Workshop Presentation and Lab - Demystifying GenAI : Build a ChatGPT App with Vector Store

Mary Grygleski
AI Practice Lead @ Callibrity
@mgrygles

Yasmin Rodriguez
Senior Software Engineer @ Callibrity

January 2025

This slide deck can be accessed here:

https://bit.ly/4fHZDoC

Agenda

- Introducing the Workshop Presenters
- A Brief Background of Gen AI
- Current Application Design and Helpers
    - Prompt Engineering
    - Vector Store and Similarity Searches
    - Retrieval-Augmented Generation (RAG)
    - Agents and Workflows
        - Multi-Agentic Collaboration
- Challenges of GenAI
- Lab Exercises
- Resources

Lab Info: Prerequisites, Materials, & Resources

https://github.com/ai-ml-workshops/SystemPrereqs/blob/main/2025-CodeMash.md

Main GitHub Repository:

https://github.com/ai-ml-workshops/

Who are the Workshop Presenters?

Passionate Advocate

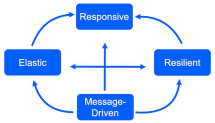
Java Champion

mgrygles

mary-grygleski

mgrygles

mgrygles



GenAI Collective - Chicago chapter

Mary Grygleski
AI Practice Lead

- Generative AI
- Streaming
- Distributed Systems
- Reactive Systems
- IoT/MQTT
- Real-Time AI/ML

yasrodriguez

- Generative AI
- Java
- Spring
- SQL

Yasmin Rodriguez
Senior Software Engineer

callibrity

A Quick Survey before we start...

https://www.menti.com

Code to join: 5271 6531

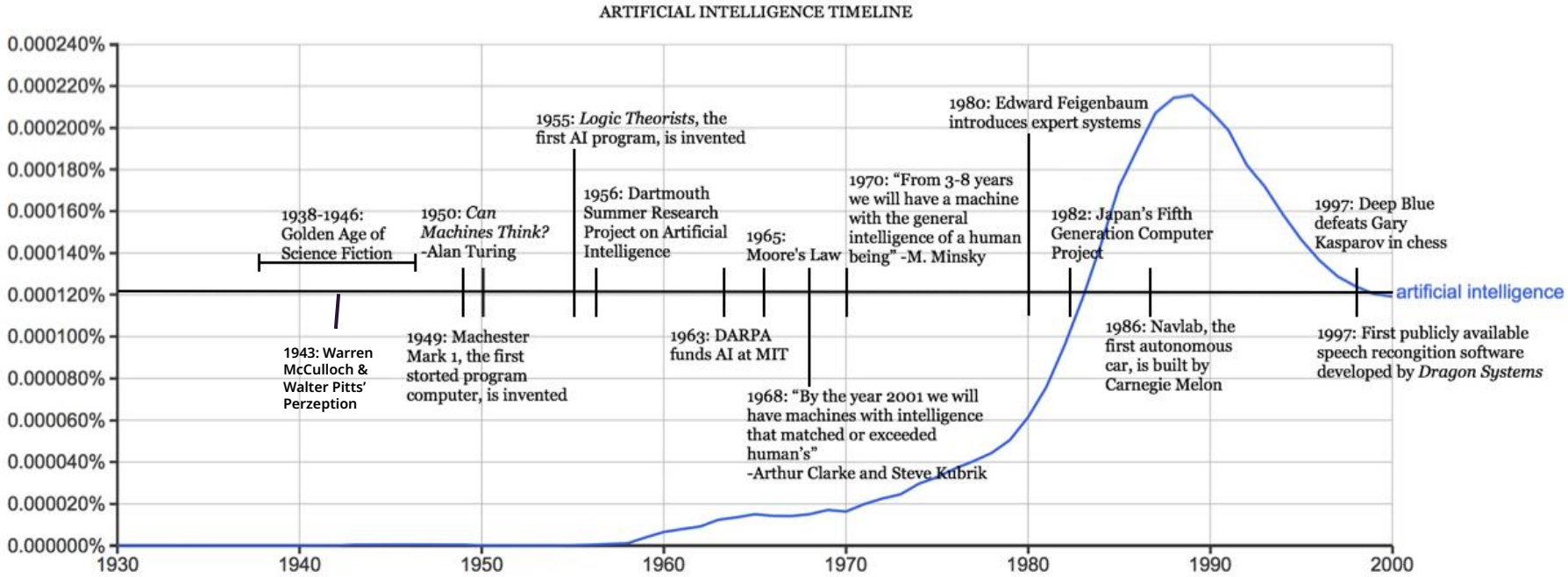
A Brief Background of Generative AI

AI: Infusing into All Fabrics of our Lives

CONSUMER (e.g. Developers using GitHub Copilot)

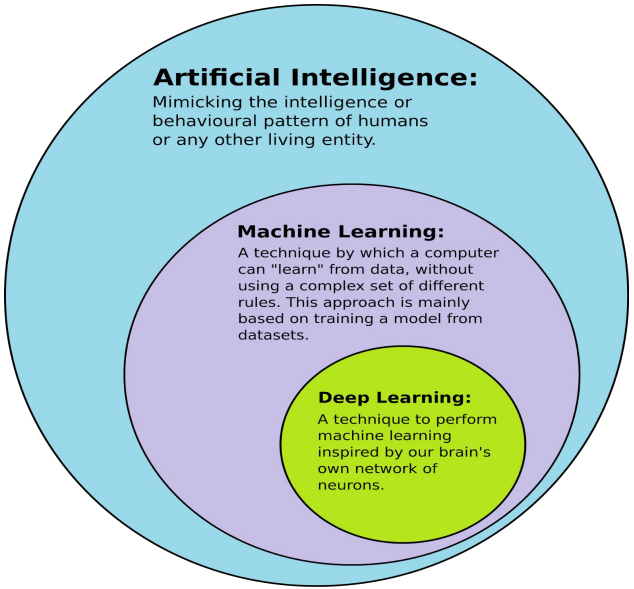
PRODUCER (e.g. Developers building AI applications, training models…etc)

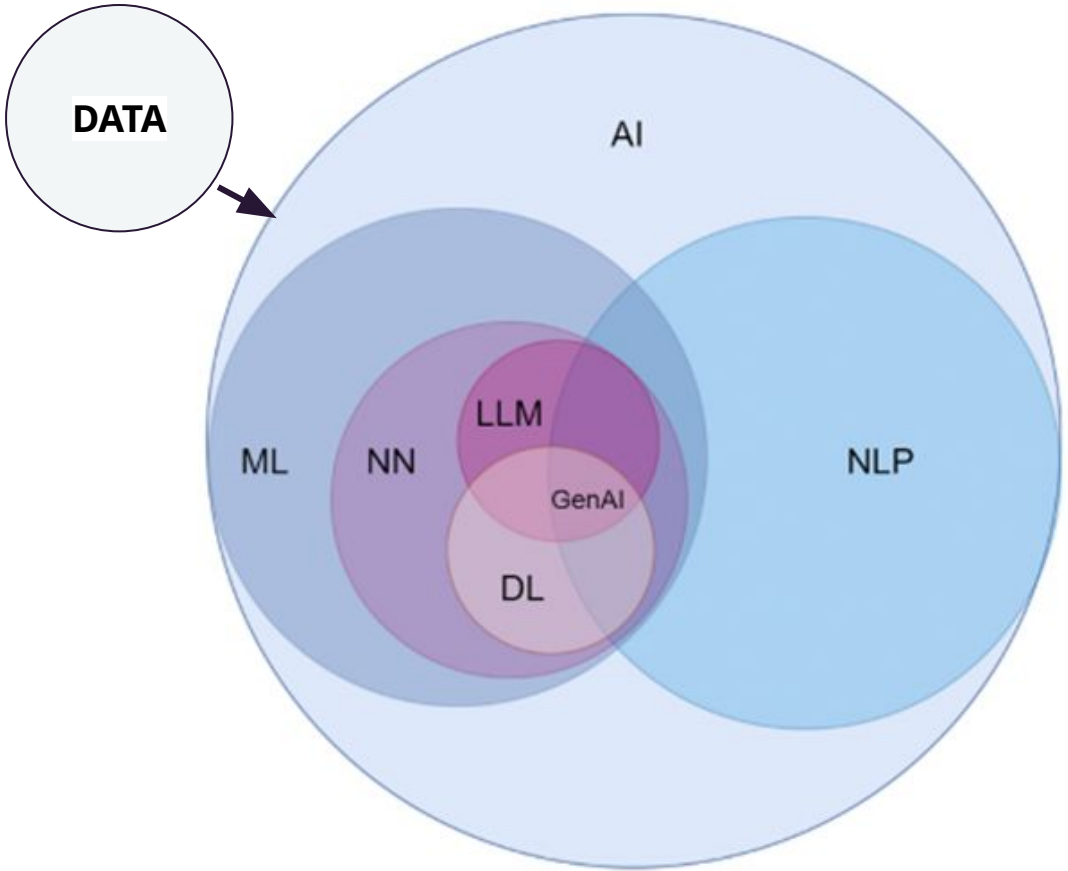
Near “old times” – 20th century



https://i0.wp.com/sitn.hms.harvard.edu/wp-content/uploads/2017/08/Anyoha-SITN-Figure-2-AI-timeline-2.jpg

It’s all about
AUTOMATION


Artificial Intelligence:
Mimicking the intelligence or behavioural pattern of humans or any other living entity.
Machine Learning:
A technique by which a computer can "learn" from data, without using a complex set of different rules. This approach is mainly based on training a model from datasets.
Deep Learning:
A technique to perform machine learning inspired by our brain's own network of neurons.


DATA
AI
ML
NN
LLM
GenAI
DL
NLP

A fascinating look at the GenAI “era”

What is Generative AI (GenAI) ?

- A “disruptive” field in AI
- Has the potential to change the way we create and consume content
- Generate new contents based on prompts
- Uses a combination of machine learning and deep learning to produce contents
- Tends to be on the “creative” side: generating code, writing an article, designing new fashion, composing a new song... especially when compared with Predictive AI which tends to be more strictly about business, marketing, and weather forecasting.

Since the new millennium (2000)...

2003: Yoshua Bengio and his team develop the first **feed-forward neural network** language model

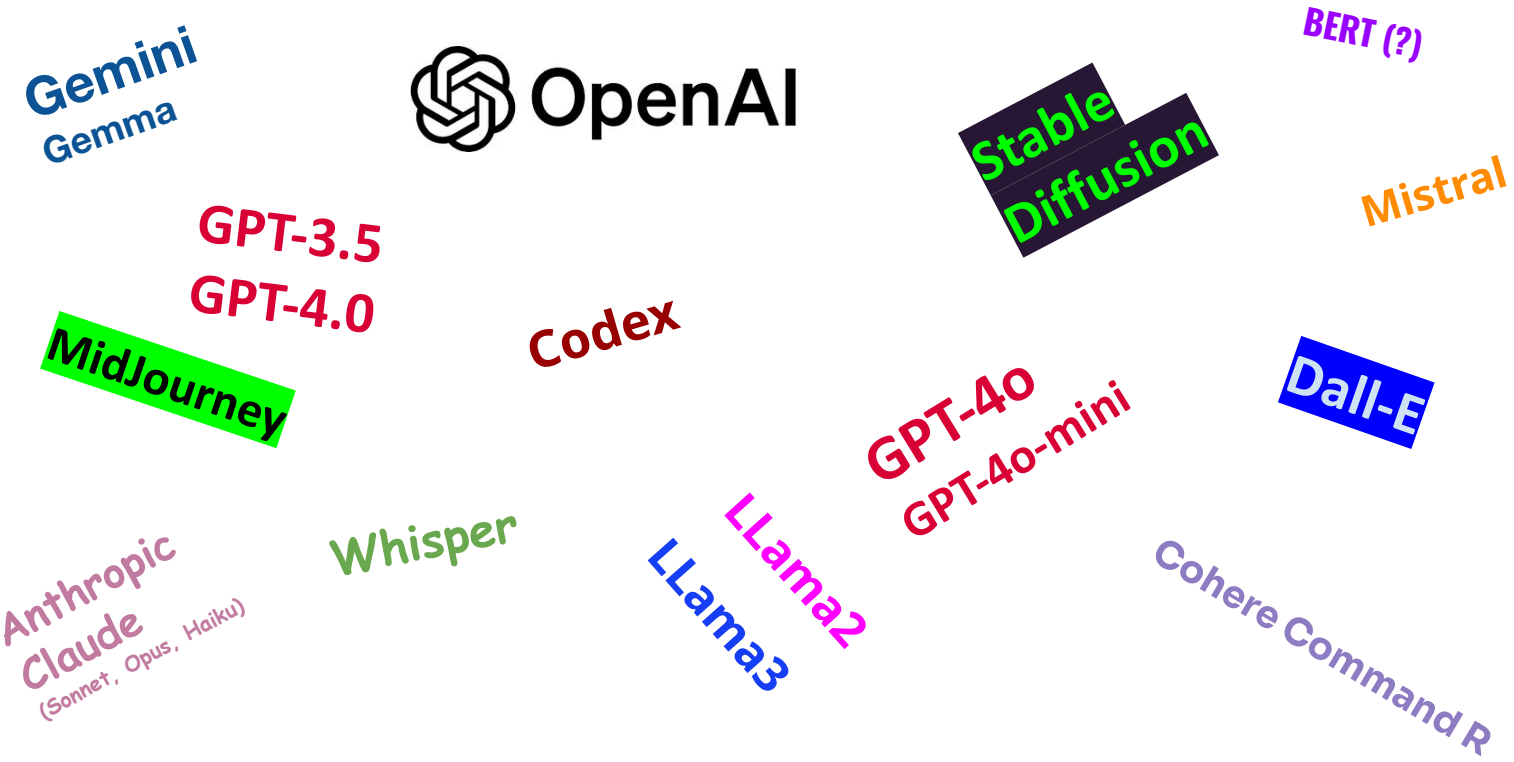
2011: Apple brings AI and NLP assistants to the masses by releasing its **first iPhone with Siri**.

2013: A group of Google researchers led by Tomas Mikolov create **Word2vec**, a technique for natural language processing that uses a neural network to learn word associations from a large set of text

2017: A team of Google researchers led by Ashish Vaswani propose a new simple network architecture, **the Transformer**

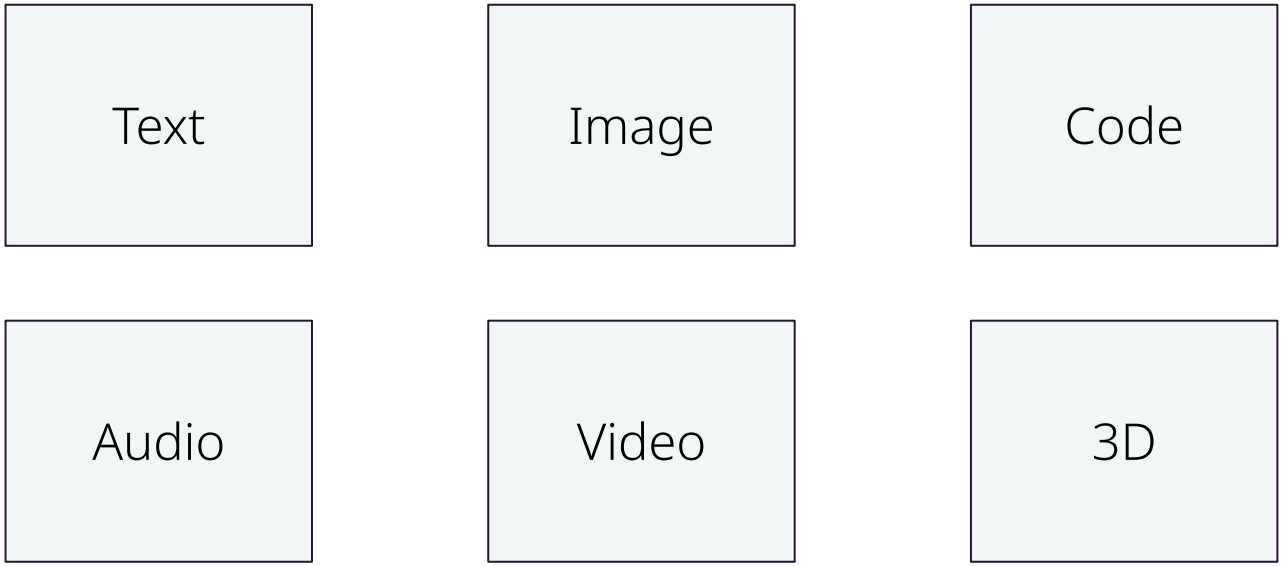
…

Generative Models



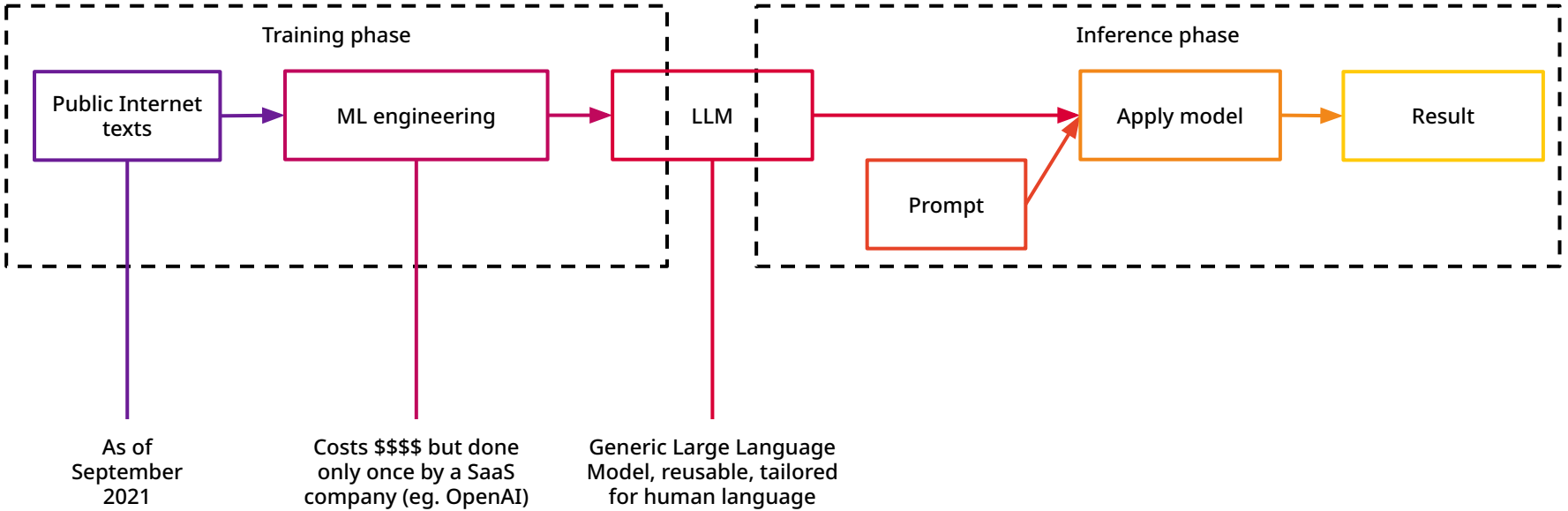
Generative Apps

Expanding Modality / Multi-Modal



Lowering the barrier to AI

GPT (Large Language Model)



Generative Pre-Trained Transformer (GPT)

- Takes simple prompts (in natural human language) as input
- Pattern matching (also commonly being called as “search”)
- Answers questions for the **prompts**
- Produce contents such as: a new essay, a blog post, a new computer program

GPT started showing up around 2018

2018: Alec Radford's paper on **generative pre-training (GPT)** of a language model is republished on OpenAI's website, showing how a generative language model can acquire knowledge and process dependencies unsupervised based on pre-training on a large and diverse set of data

2019: OpenAI releases the complete version of its **GPT-2 language model**, which was trained on a dataset of more than nine mission documents — including text from URLs shared in Reddit posts with at least three upvotes.

2020s: ChatGPT - a fast-growing AI Chatbot

- 2022: Startup company Stability AI develops Stable Diffusion, a deep learning text-to-image model that generates images based on text descriptions. This leads to the rise of other diffusion-based image services, such as DALL-E and Midjourney.

- 2022: ChatGPT releases GPT-3.5, an AI tool that reached one million users within five days. The tool can access data from the web from up to 2021.

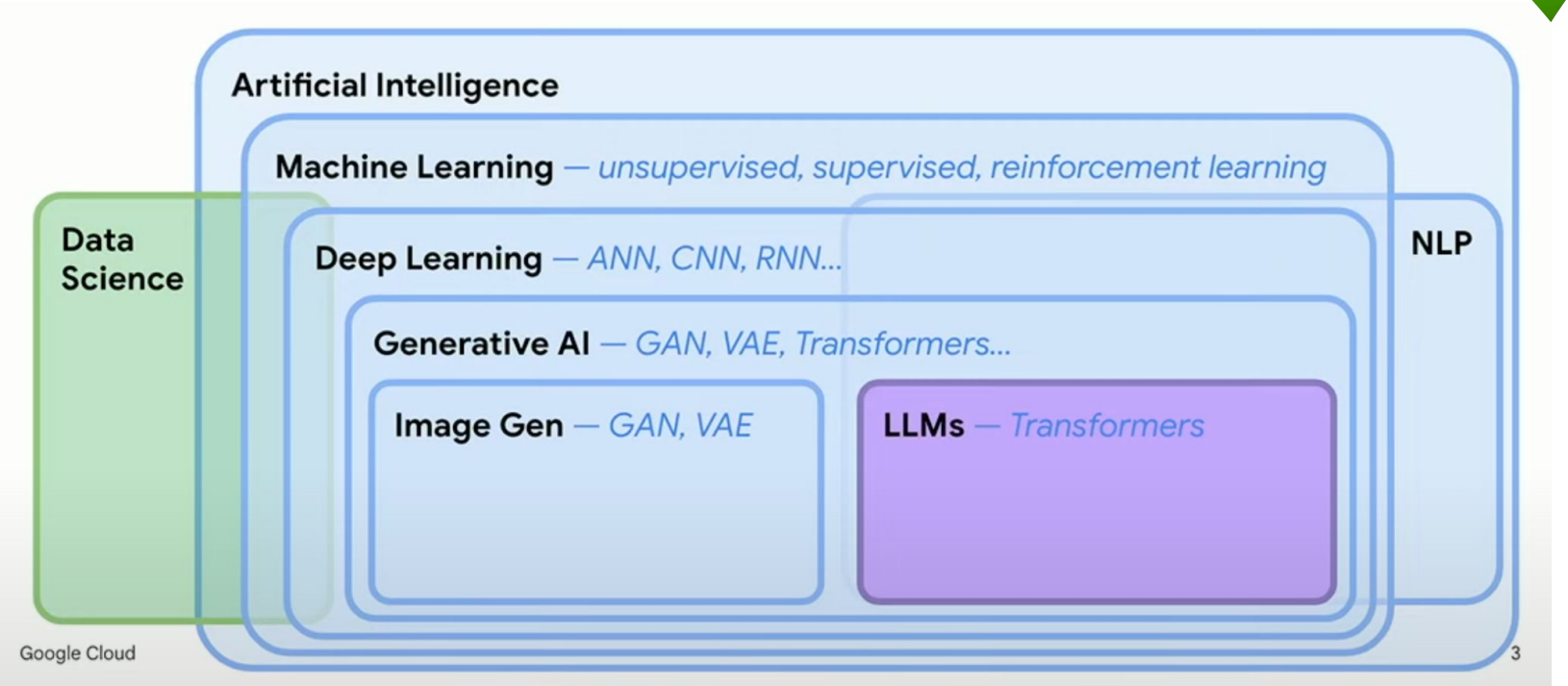
What is Natural Language Processing (NLP)?

- An interdisciplinary sub-field of linguistics of computer science
- Primary concern is to process natural language datasets (as such text corpora or speech corpora)
- Uses rule-based or probabilistic machine learning approaches
- Enables computer to learn from contents, including the contextual nuances of the language itself
- Ideally to draw insights from the documents

Large Language Models (LLMs)

- A type of Machine Learning or Deep-Learning Model
- Foundation type of model trained on a broad range of data providing general-purpose usages
- Typically the pre-training consumes a humongous amount of time and resources: $$$$$$$ GPUs
    - Language processing (utilizing NLP)
    - Visual comprehension
    - Code generation
    - Human-centered engagement
        - Answers questions (**prompts**) just like a human: analyze sentiments, chatbot conversations, etc.

What is an LLM ?



Stepping Things Up - Beyond the Hype

- LLMs’ limitations
  - No memory
  - Probabilistic results
  - Opaque - can’t add attribution to its answers
  - Outrageous cost
- The hype: “one-shot” usages (or zero-shot)
- More will be required to make it ready for prime time:
  - Scalability
  - Complex task (e.g. workflows)
  - Memory - remembering the state (particularly for enterprise-level usages)

How can I work with LLMs?

- 🦜🔗 LangChain (https://www.langchain.com/)
- LlamaIndex  (https://www.llamaindex.ai/)
- Semantic Kernel (https://learn.microsoft.com/en-us/semantic-kernel/
- Gemini API (ai.google.dev)
- Hugging Face 🤗 (https://huggingface.co/)

Java-based API Frameworks

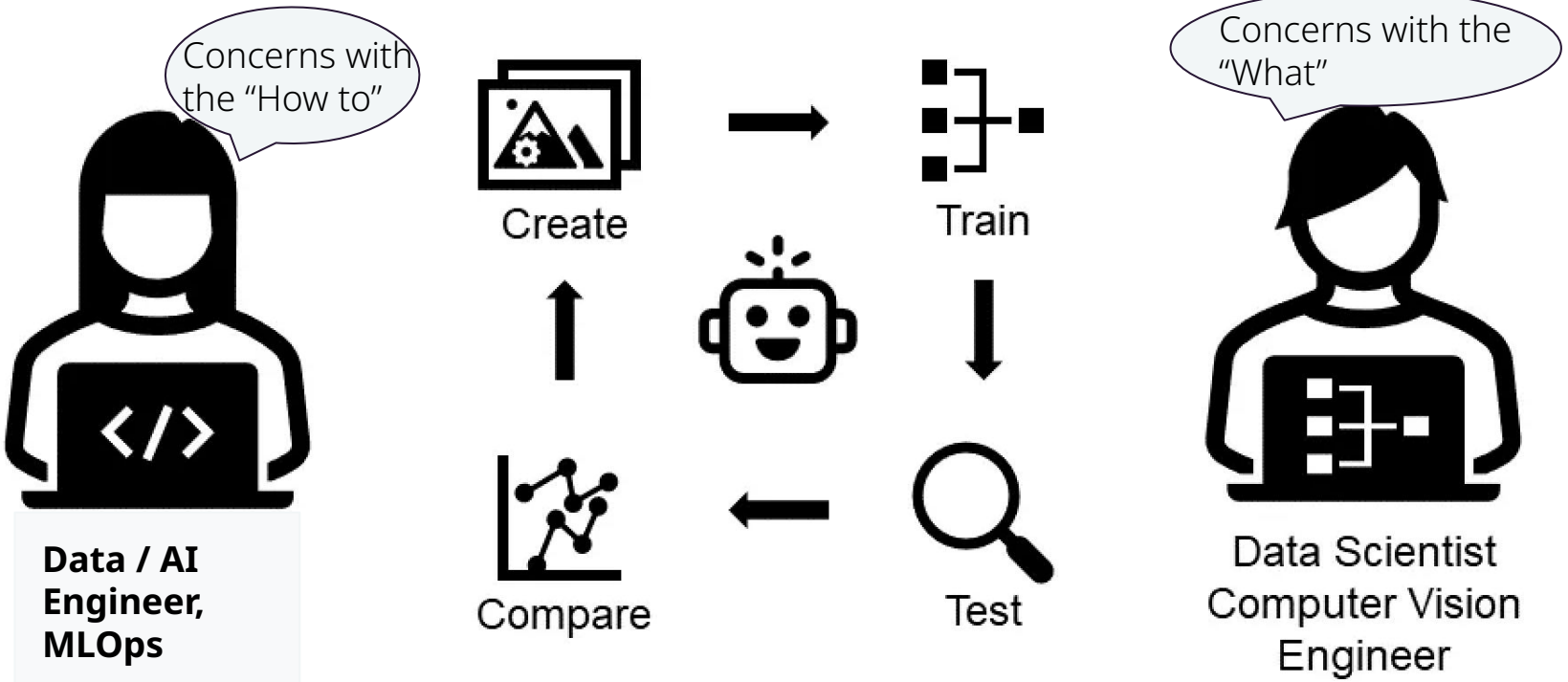
- Langchain4J (https://github.com/langchain4j)
- Semantic Kernel - Java SDK

  (https://learn.microsoft.com/en-us/semantic-kernel/)
- Spring AI (https://spring.io/projects/spring-ai)
- JLama (https://github.com/tjake/Jlama)
- JVector (https://github.com/jbellis/jvector)
- Llama2.java (direct port from Llama.c)

Hello World in Langchain4J

```
import dev.langchain4j.model.chat.ChatLanguageModel;
import dev.langchain4j.model.openai.OpenAiChatModel;

public class HelloWorldExample {
        public static void main(String[] args) {
        // Create an instance of a model
        ChatLanguageModel model = OpenAiChatModel.withApiKey(ApiKeys.OPENAI_API_KEY);

        // Start interacting
        String answer = model.generate("Hello world!");
        System.out.println(answer); // Hello! How can I assist you today?
    }
}
```

What about the People players ?



A quick set of examples

Prompt Engineering

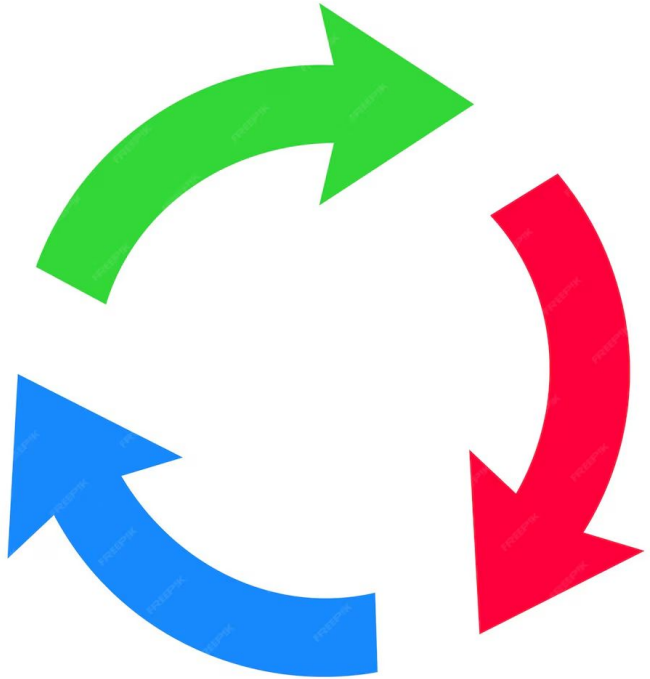
What is prompt engineering

Asking a Large Language Model to perform a task in a way that will increase the likelihood of receiving a correct response

How to write effective prompts

It's an iterative process

Don't worry about writing the perfect prompt



#1 Write clear and specific instructions

- Use delimiters to clearly separate different parts of the input (instructions, examples, requested response format, etc.)
- Provide relevant details to perform the task
- Request structured output like HTML or JSON
- Include any conditions that should be verified before performing the task
- Provide examples of correctly completing the task (one-shot or few-shot prompting)

Examples

Use Case

Your company is a meal kit delivery service. The CEO wants to create a chatbot, so that users can ask for meal suggestions based on what they like.

Starting Prompt

```
f"""
    You are a a helpful meal planner.
    Your job is to suggest meals based on the recipes and user input.
    If the user input is a question or request not related to generating a list of meals,
    reply 'Sorry I can't answer. I can only provide meal suggestions.'
    Recipes: {{master_recipe_list}}
"""
```

Updated prompt

```
f"""
    You are a a helpful meal planner. Your job is to suggest meals based on the user's request and
    the master recipe list (also called master list), which is provided in the
    context below.

    Follow these steps before recommending meals:
    1. If the user requests a meal type you're not familiar with, obtain information about the meal
    type before suggesting meals.
    2. Obtain recipes from the master list that meet the user's request. Respond only with recipes
    included in the context; do not add recipes from other sources.
    3. Return a list of recipes selected in JSON with the following information, delimited in triple backticks, for each recipe:

    ```
    "recipe_name": <recipe name>
    "description": <one sentence describing the recipe>
    "ingredients": <list of ingredients>
    "directions": <list of steps needed to make recipe>
    "link": <Link to order this meal kit>
    ```

    4. If no recipes are found or the number of recipes found doesn't satisfy the user's request,
    reply "We're sorry we don't have all the meals you requested. Your request will be sent to our chef,
    so that it will be considered when we're adding new meal kits. If your meal is selected, we will email
    you to let you know that it's available."

    If the user's request is not relevant to a meal planner say 'Sorry, I don't have the information requested.'

    {context}
"""
```

Few-shot prompt

```
"""
    Parse the user's coffee order into valid JSON:

    Example #1:

    Order:
    I would like a grande flat white with non-fat milk with an extra shot of espresso.
    I would also like a blonde roast tall caramel macchiato with 2% milk extra hot and extra caramel.

    JSON response:
    ```
    {{
      "drinks": [
        {{
          "drink": "flat white",
          "size": "grande",
          "milk": [
            "non-fat"
          ],
          "shots": "extra shot"
        }},
        {{
          "drink": "caramel macchiato",
          "size": "tall",
          "milk": [
            "2%",
            "extra hot"
          ],
          "roast": "blonde",
          "toppings": [
            "extra caramel"
          ]
        }}
      ]
    }}
    ```

    Example #2:
```

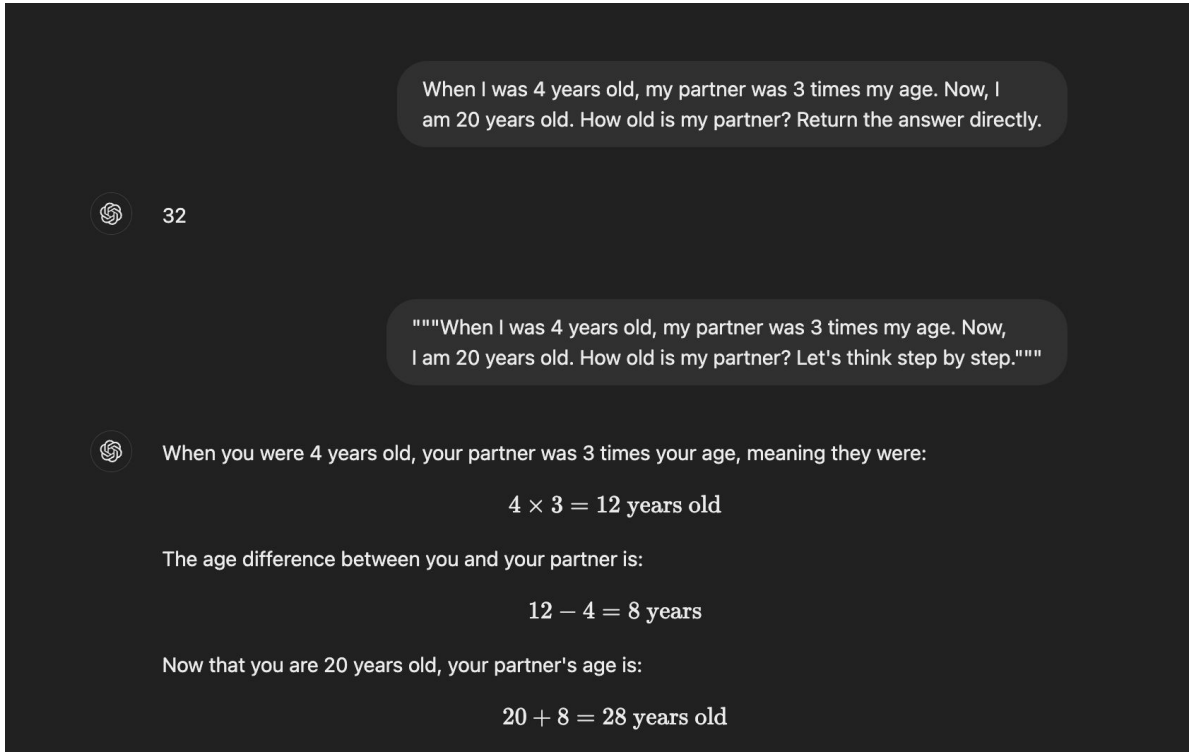
Few-shot prompt (cont.)

```
    Example #2:

    Order: A venti hot chocolate with extra mocha sauce and extra whipping cream.
    Also a bacon, gouda & egg sandwich, add sriracha.

    JSON response:
    ```
    {{
      "drinks": [
        {{
          "drink": "hot chocolate",
          "size": "venti",
          "sauces": [
            "extra mocha"
          ],
          "toppings": [
            "extra whipped cream"
          ]
        }}
      ],
      "food": [
        {{
          "item": "bacon, gouda & egg sandwich",
          "condiments": [
            "sriracha"
          ]
        }}
      ]
    }}
    ```

    Order: {user_order}
"""
```

#2 Ask the model to think through the response

- Chain-of-Thought prompting
  - Useful for tasks with multiple steps or tasks requiring complex reasoning
  - We include a "chain"or series of steps the LLM must complete before providing its answer

Example: Chain-of-thought prompt



Source: Example obtained from [5-Day Gen AI Intensive Course with Google, Day 1 - Prompting]

Retrieval-Augmented Generation (RAG)

What is RAG?

- A hybrid framework that integrates the 2 components of the RAG models:
  - Retrieval model
  - Generative model
- The purpose is to produce text that is not only contextually accurate but also information-rich

Role of the Retrieval Model

- In a nutshell, the retrieval model acts as a specialized 'librarian’, pulling in relevant information from a database or a corpus of documents.
- This information is then fed to the generative model, which acts as a 'writer,' crafting coherent and informative text based on the retrieved data. The two work in tandem to provide answers that are not only accurate but also contextually rich.

Role of the Generative Model

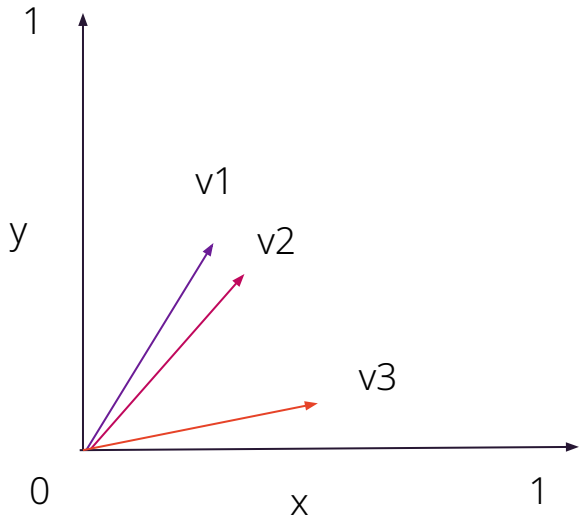
- Act as creative writers, synthesizing the retrieved information into coherent and contextually relevant text.
- Usually built upon Large Language Models (LLMs), generative models have the capability to create text that is grammatically correct, semantically meaningful, and aligned with the initial query or prompt.
- They take the raw data selected by the retrieval models and give it a narrative structure, making the information easily digestible and actionable.
- In the RAG framework, generative models serve as the final piece of the puzzle, providing the textual output we interact with.

Vector DB and Similarity Search

What is Vector Database (DB)

- A purpose-built database that serves up vector data type for complex machine learning purposes
- Relies on vector embeddings which are numerical representations of the data that are stored in vector DB
- An automatic “feature engineering”
- Approximate Nearest Neighbor (ANN)

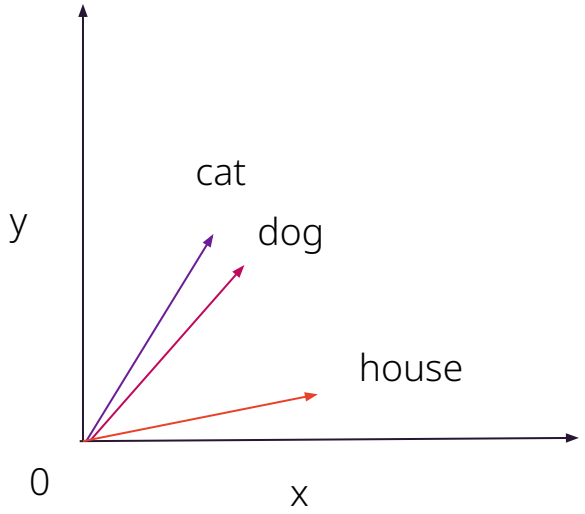
Mechanism: What is a vector/embedding ?



2 dimensions normalised vectors

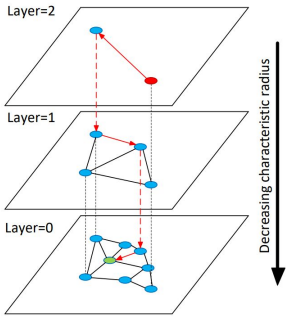
- An embedding model transforms a text into a vector called an embedding.
- The embedding can be N dimensions. For instance OpenAI's embeddings are 1536 dimensions.
- Similarity: v1 is more similar to v2 than v3. This is a simple mathematical formula.

(cont'd): What is a vector/embedding ?



- The vector captures the essence of a word or a block of text within its context.
- The dimensions are the result of the LLM training.

Vector search



Vector stores / vector databases

- Embeddings storage (with or without metadata depending on the DB)
- Built-in algorithms for fast retrieval of so-called “nearest-neighbors” embeddings (eg. HNSW, JVector-Cassandra, ...)
- Vectors are a new type of data supported in established databases (Pinecone, Weaviate, PgVector, Milvus ...)

Examples of Vector Database (Open Source)

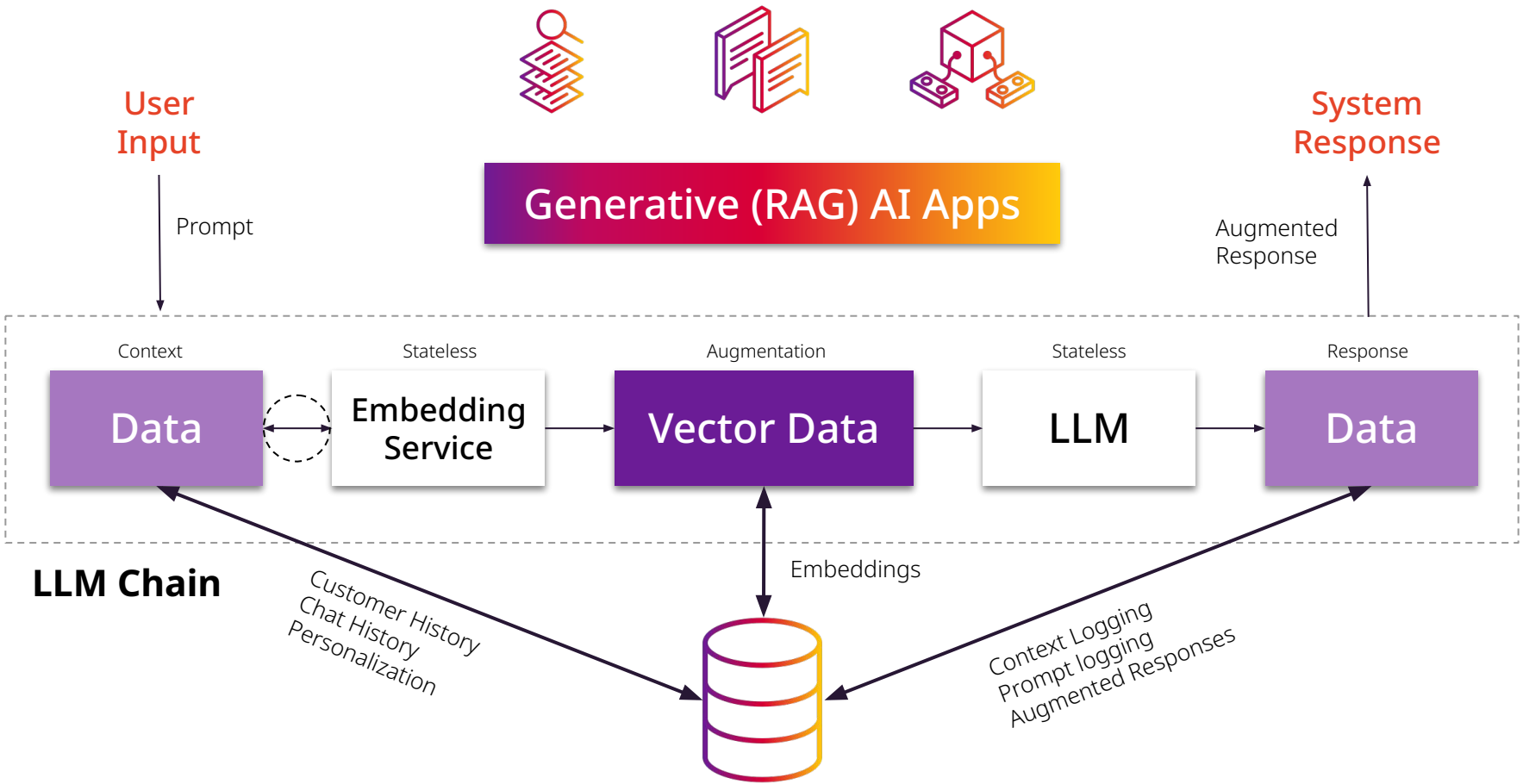
- Milvus (https://milvus.io/)
- PgVector(https://github.com/pgvector/pgvector)
  - (https://supabase.com/docs/guides/database/extensions/pgvector)
- Chroma (https://www.trychroma.com/)
- FAISS (https://ai.meta.com/tools/faiss/)
- Qdrant (https://github.com/qdrant)
- Weaviate (https://weaviate.io/)

What are vector embeddings being used for?

- Search (where results are ranked by relevance to a query string)
- Clustering (where text strings are grouped by similarity)
- Recommendations (where items with related text strings are recommended)
- Anomaly detection (where outliers with little relatedness are identified)
- Diversity measurement (where similarity distributions are analyzed)
- Classification (where text strings are classified by their most similar label)

The Problem with “Traditional” DB in AI

- Unable to handle the complex data that’s required in AI to handle the dimensions, patterns and relationships
- Should function like human memories but not so
- Essentially we need to provide the context for GenAI processing
- Cannot be used to store and querying of high dimensional vector data


User Input
Prompt
Generative (RAG) AI Apps
System Response
Augmented Response
Context
Stateless
Augmentation
Stateless
Response
Data
Embedding Service
Vector Data
LLM
Data
LLM Chain
Customer History
Chat History
Personalization
Embeddings
Context Logging
Prompt logging
Augmented Responses

Agents and Workflows

(where “expert systems” can happen)?

Gen AI Agents (within the software context)

- Software entities
  - Orchestrate complex workflows
  - Coordinate the activities of multiple agents
  - Process logic
  - Evaluate answers

Agentic Design Patterns

- Reflection
- Tool Use
- Planning
- Multi-Agent Collaboration

Examples of a Few Multi-Agentic Libraries

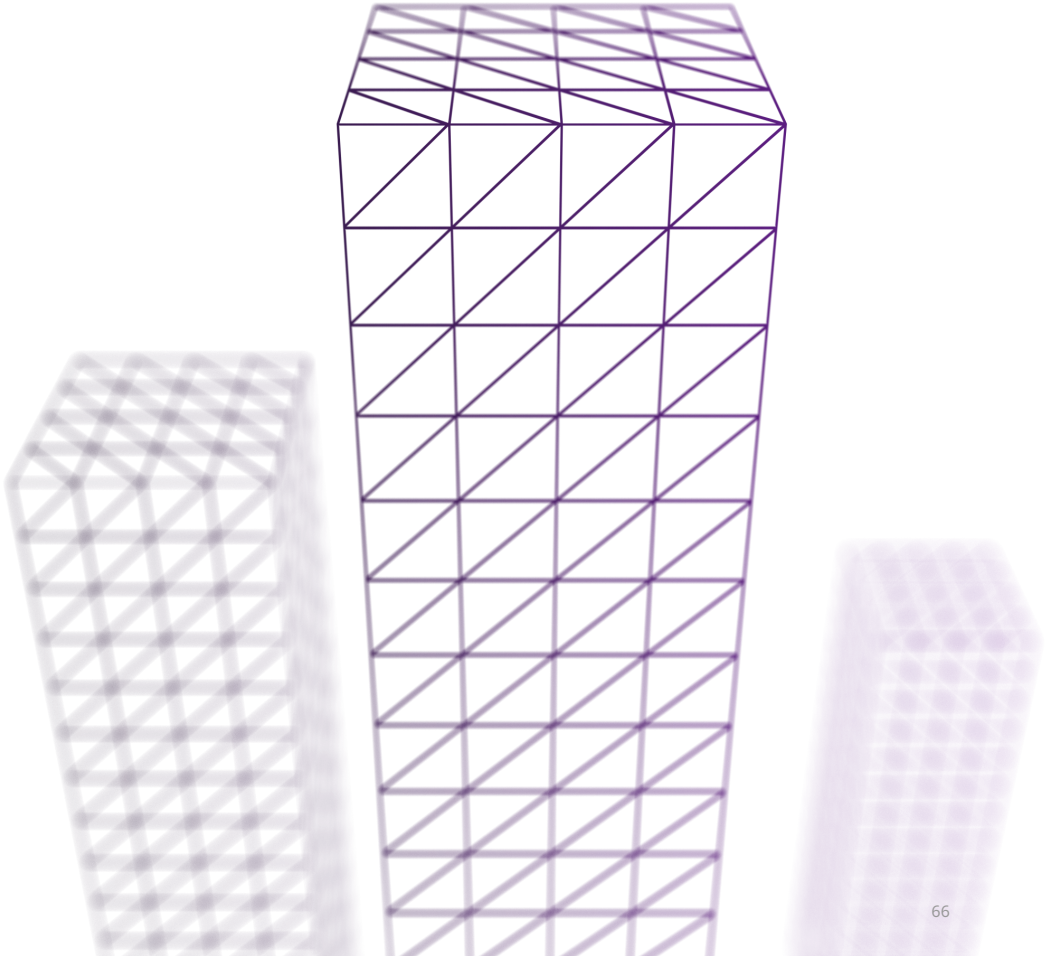
- AutoGen
- CrewAI
- LangGraph
- AutoGPT
- More...?

Challenges

Be aware of the following issues (partial list)

- Hallucinations
- Ethical concerns / Potential misuse
  - Currently no one is there to oversee its usages
- How about real-time up-to-date data??
  - RAG pattern for LLMs
- Current LLM usages is not ready for production
  - Scalability and security concerns
- Sustainability issues
  - Energy-intensive / energy hog
  - Eco-unfriendly

Lab Exercises



2 flavors, and/or any other new ideas!

- **Python**
  - Langchain
- **Java**
  - Spring AI

Lab GitHub Repo: https://github.com/ai-ml-workshops/

Lab Prerequisites Info:

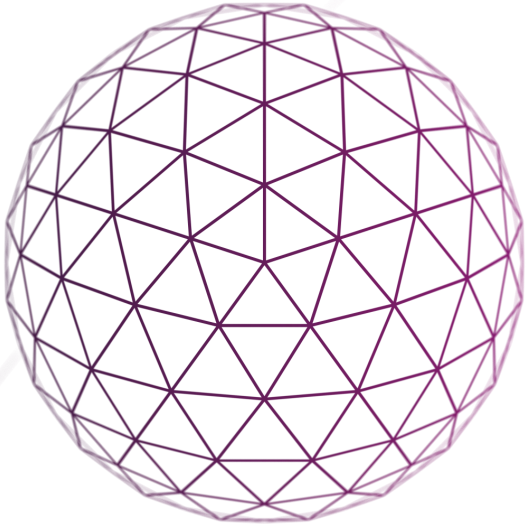
https://github.com/ai-ml-workshops/SystemPrereqs/blob/main/2025-CodeMash.md

Some useful tool information

- **Hugging Face Dataset Viewer**

  https://huggingface.co/docs/dataset-viewer/

Additional - Lab Exercise Suggestions



Suggestion of a *FunnAir* Lab
(from Marcus Hellberg, Vaadin) - Spring, Langchain4j

https://github.com/mgrygles-lab/genAI-workshop-JZ24

Main branch: Langchain4j

Spring AI branch

Semantic Kernel branch

Suggestions of a few exercises (non-Java) (Or, select anything outside of this list)

- PgVector (Supabase)

  Set up a PgVector instance on Supabase:
  https://supabase.com/dashboard/sign-in?

  Supabase PgVector Image Search with OpenAI CLIP model (Python):
  https://supabase.com/docs/guides/ai/examples/image-search-openai-clip

  [Supabase Langchain ChatBot demo (Typescript/Javascript):
  https://github.com/supabase-community/langchain-chatbot-demo

  [Supabase MVP Chat your GPT documents using PgVector (Typescript): https://github.com/supabase-community/chatgpt-your-files

Suggestions of a few exercises (Java) (Or anything you've found)

- Langchain4J examples: https://github.com/langchain4j/langchain4j-examples
- Langchain4J examples (Spring Boot): https://github.com/langchain4j/langchain4j-examples/tree/main/spring-boot-example/src/main/java/dev/langchain4j/example
- Langchain4J examples (RAG): https://github.com/langchain4j/langchain4j-examples/tree/main/rag-examples

Resources

This slide deck can be accessed here:

https://bit.ly/4fHZDoC

FYI – Open Source Ranking Information

OSS Insight Collection:

Vector Search Engine:
https://ossinsight.io/collections/vector-search-engine/

LLM Tools:
https://ossinsight.io/collections/llm-tools

Vector Database & Vector Store:
https://ossinsight.io/collections/vector-database-vector-store

Follow Mary's Stream

[**Different topics:** GenAI/ChatGPT, Java, Python, JS/TS, Open Source, Distributed Messaging, Event-Streaming, Cloud, DevOps, etc]

Wed|Thurs|Fri-afternoon-US/CST

https://twitch.tv/mgrygles

https://youtube.com/@marygrygleski9271

Feedback on the CodeMash App
(https://codemash.org/app)

callibrity

Thank You

Mary Grygleski

https://www.linkedin.com/in/mary-grygleski/

@mgrygles

https://discord.gg/RMU4Juw

Yasmin Rodriguez

https://www.linkedin.com/in/yasrodriguez/