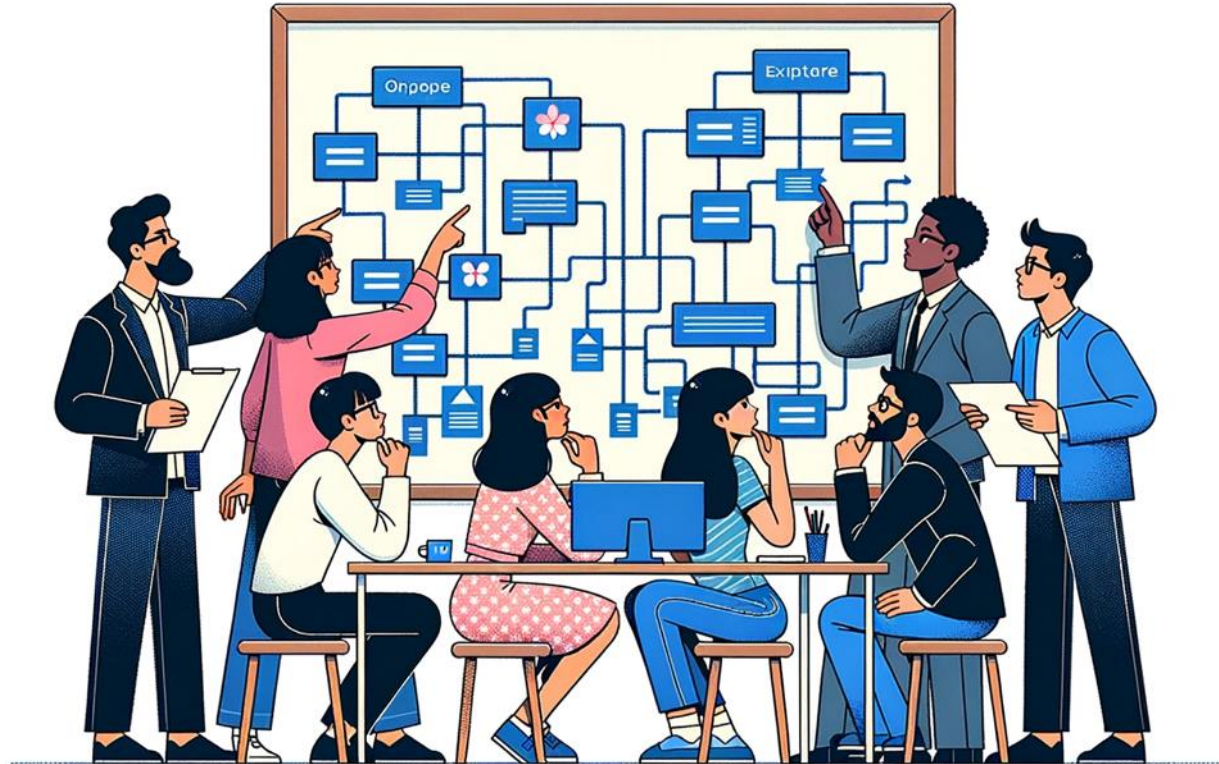


# Generative Patterns for AI



Foundations of AI Acceleration

Dan McCreary and Steve Peterson

November 2023

# Talk Description

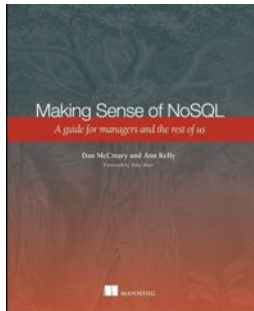
---

Generative AI is by far the most impactful technology in the last ten years. It dominates the priority list of almost every CIO in the world.

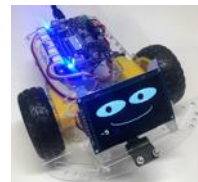
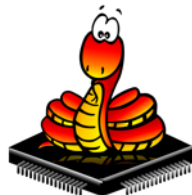
This session will review key architectural patterns that organizations are using to safely harness the power of Generative AI to build intelligent search, chatbots and agents as well as performing a variety of tasks on unstructured text.

We will cover the key use case taxonomies that are emerging and the architectural patterns we need to support these initiatives. If implemented, these patterns can accelerate the process of building intelligent agents for all knowledge workers.

# Background for Dan McCreary



- Solution architect with over 35 years working with data
- Worked for Bell Labs and Steve Jobs at NeXT
- Author of a book on database selection “Making Sense of NoSQL”
- Promoter of XML and graph query standards (GQL)
- Former Distinguished Engineer working for Optum (UHG)
- Built the worlds largest healthcare knowledge graph
- Lead the “Generative AI Center of Excellence” at Optum
- Worked with over 250 healthcare uses cases – 50 which were funded and over 10 in production
- Blogger: <http://medium.com/@dmccreary>
- Big fan of AI, robotics, MicroPython, and STEM education
- Advocate for neurodiversity



# Background for Steve Peterson

---



## Steve Peterson

Senior Director of Enterprise Architecture  
Rodan + Fields.

Steve brings over three decades of expertise in spearheading enterprise architecture initiatives. His career has seen pivotal roles at industry giants such as IBM, Be The Match, UnitedHealthcare (UHC) & Optum, and AT&T. At UHC, Steve's visionary leadership in Enterprise Architecture was evident as he crafted strategic roadmaps, particularly emphasizing innovations in integration, chatbots, and Natural Language Processing.

# Outline

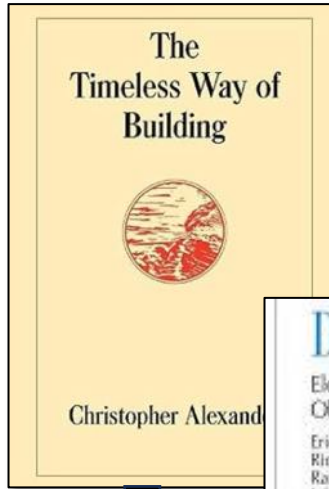
---

What  
Why  
Concepts  
Patterns  
Examples  
Resources

- **What** is an architectural pattern?
- **Why** are they important for discussing architecture tradeoffs?
- Building on Enterprise Integration Patterns (EIP)
- Key **concepts**
  - Large-language model
  - Embedding
  - Vector store
- Key **patterns**:
  - Finetuning
  - Prompt Enrichment (RAG)
  - Embeddings
  - Concept Index (vector store)

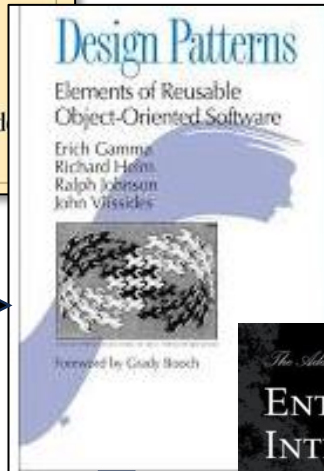
Supporting Website: <https://dmccreary.github.io/genai-arch-patterns/>

# What is an Architectural Pattern?

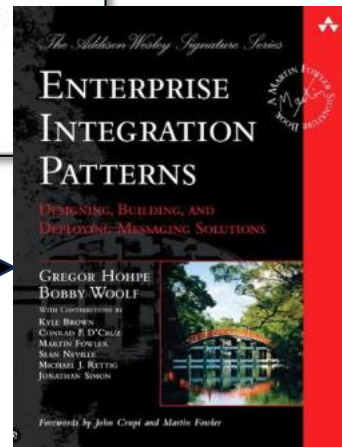


1979

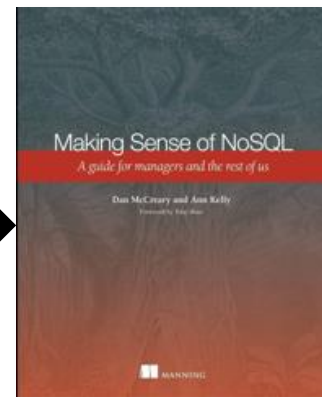
- A general, **reusable solution** to a commonly occurring problem in software architecture within a particular context
- A **named** way of moving from problem to solution



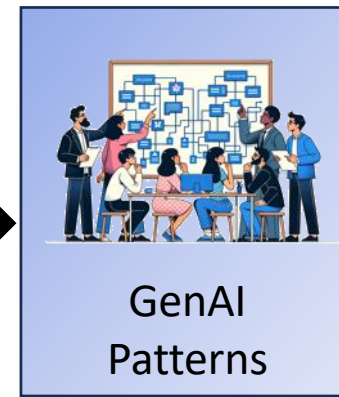
1994



2003



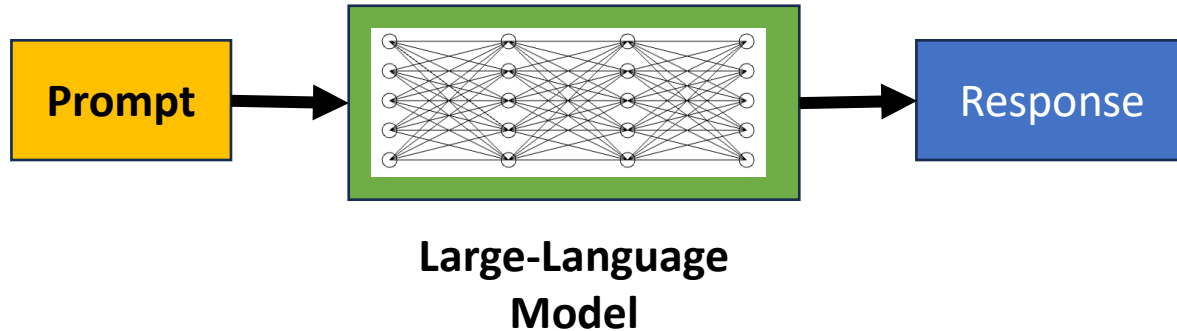
2014



2023

# Abstractions and Subsystems

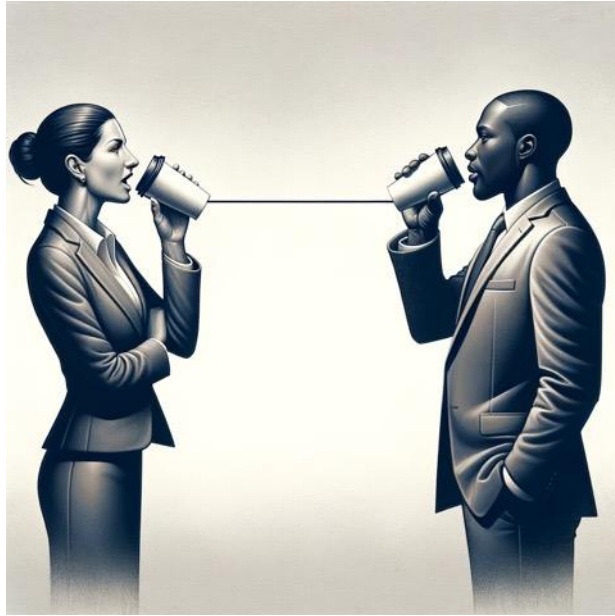
---



Architectural patterns are high-level **abstractions** that provides a set of predefined subsystems, their responsibilities, and rules for organizing the relationships between them.

# Patterns and Bandwidth

---

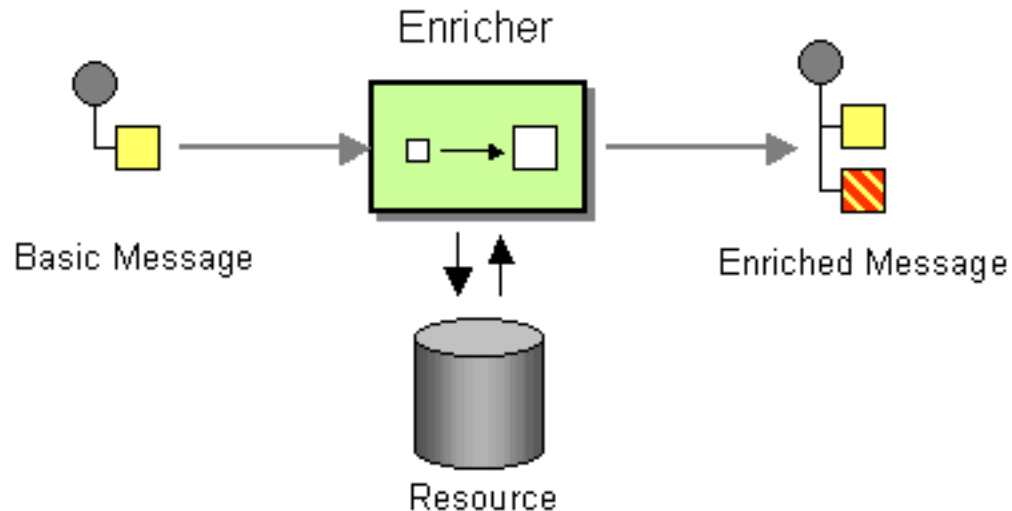


- Architecture patterns help in laying out the structure for a software system and provide a **vocabulary** for discussing system design
- Precise design patterns create high-bandwidth discussions that help architects quickly understand design tradeoffs



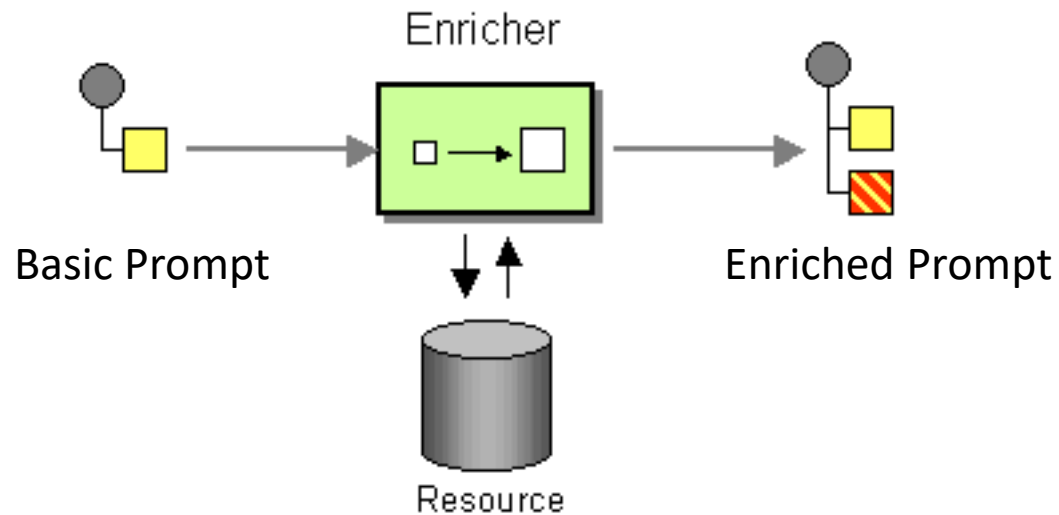
# Architecture Patterns Have Diagrams

## Enterprise Integration Patterns



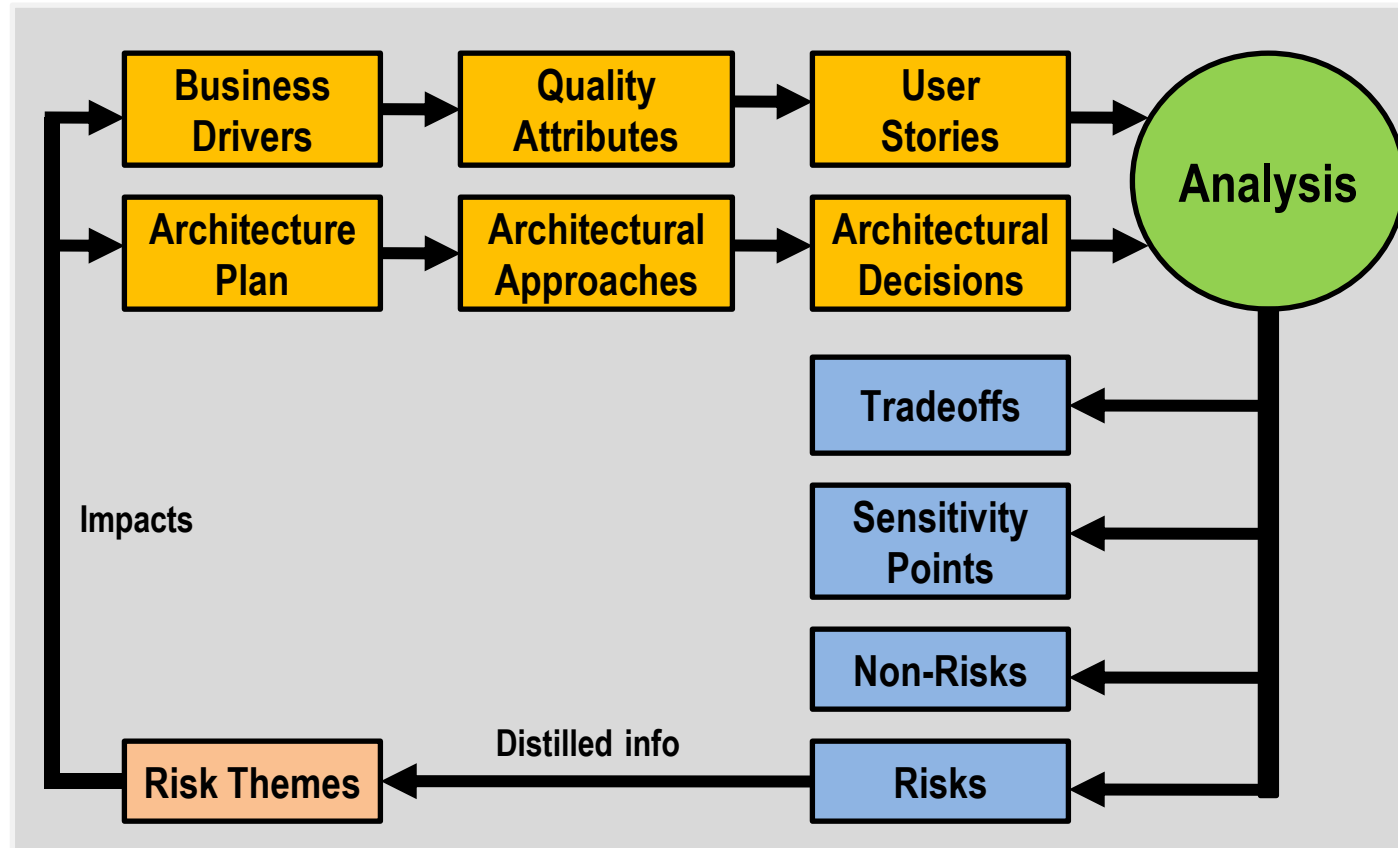
- Good architecture patterns have clear diagrams that have common meaning
- New patterns can be derived from similar patterns
- Example EIP Pattern: Content Enrichment

# Prompt Enrichment



- By using similar terminology that architects are already familiar with, we can quickly build new patterns

# Architecture Tradeoff Analysis



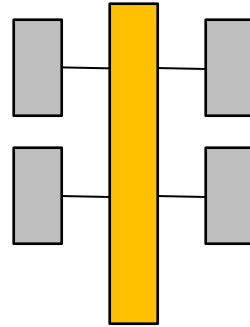
This process defined by CMU's Software Engineering Institute

# Example: Databases Architectures

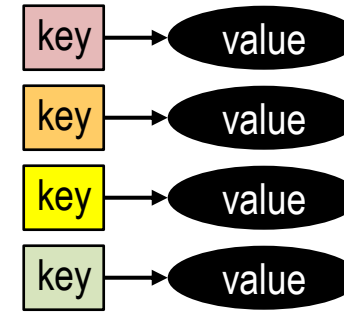
**Relational**



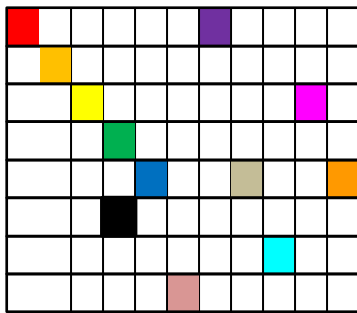
**Analytical (OLAP)**



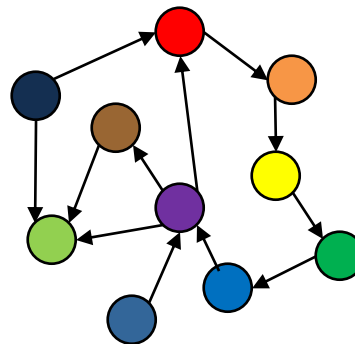
**Key-Value**



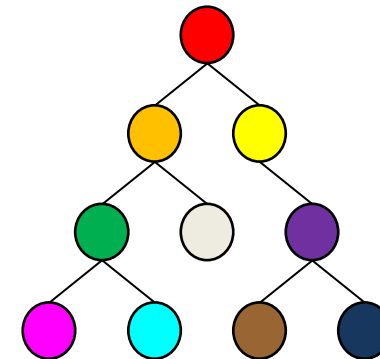
**Column-Family**



**Graph**



**Document**



There is no once-size-fits all. There are only tradeoffs.

# Architecture Tradeoffs in Generative AI

---

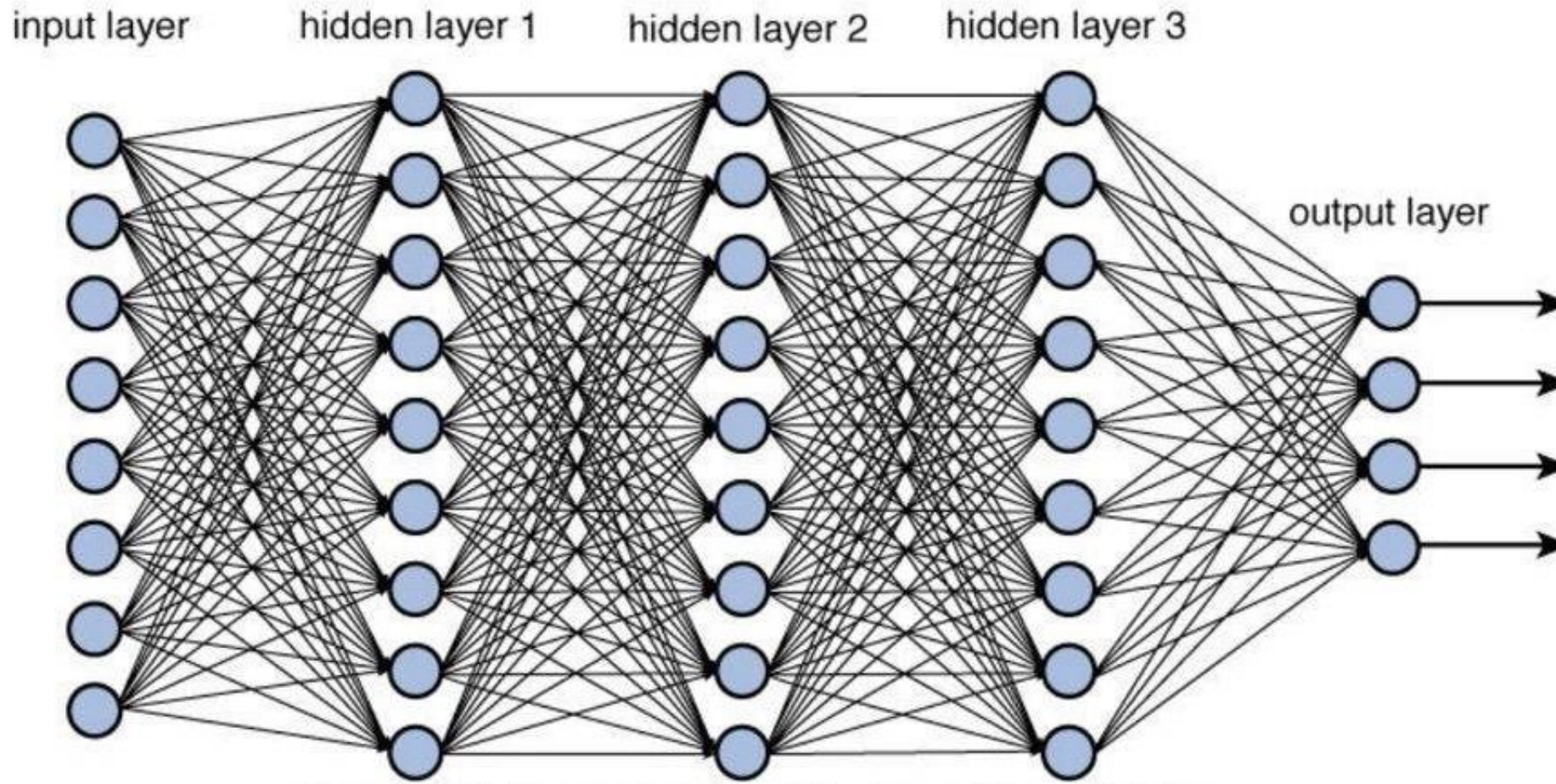
- Finetune or Prompt Enrichment?
- Small model or large model?
- Small context window or large?
- Faster response time or higher quality?
- Vector database or graph database?
- Security at the database or application level?
  - All public documents or role-based access control?

# Key GenAI Concepts

---

- Neural networks
- Large language models
- Prompts
- Tokens
- Prompt context window
- Prompt design and engineering
- Embeddings
- Concept Index (Vector database)
- Comparison and Similarity
- Classification
- Multimodal

# Deep Neural Network



- Image source:

# Comparing Items: The Geospatial Metaphor



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$d$  = distance

$(x_1, y_1)$  = coordinates of the first point

$(x_2, y_2)$  = coordinates of the second point

- Any two locations have a “distance” in space between them
- We extend this to more than two dimensions
- An embedding is expressed as a set of “coordinates” in a 200-dimensional space



# Text Embedding

---

Text

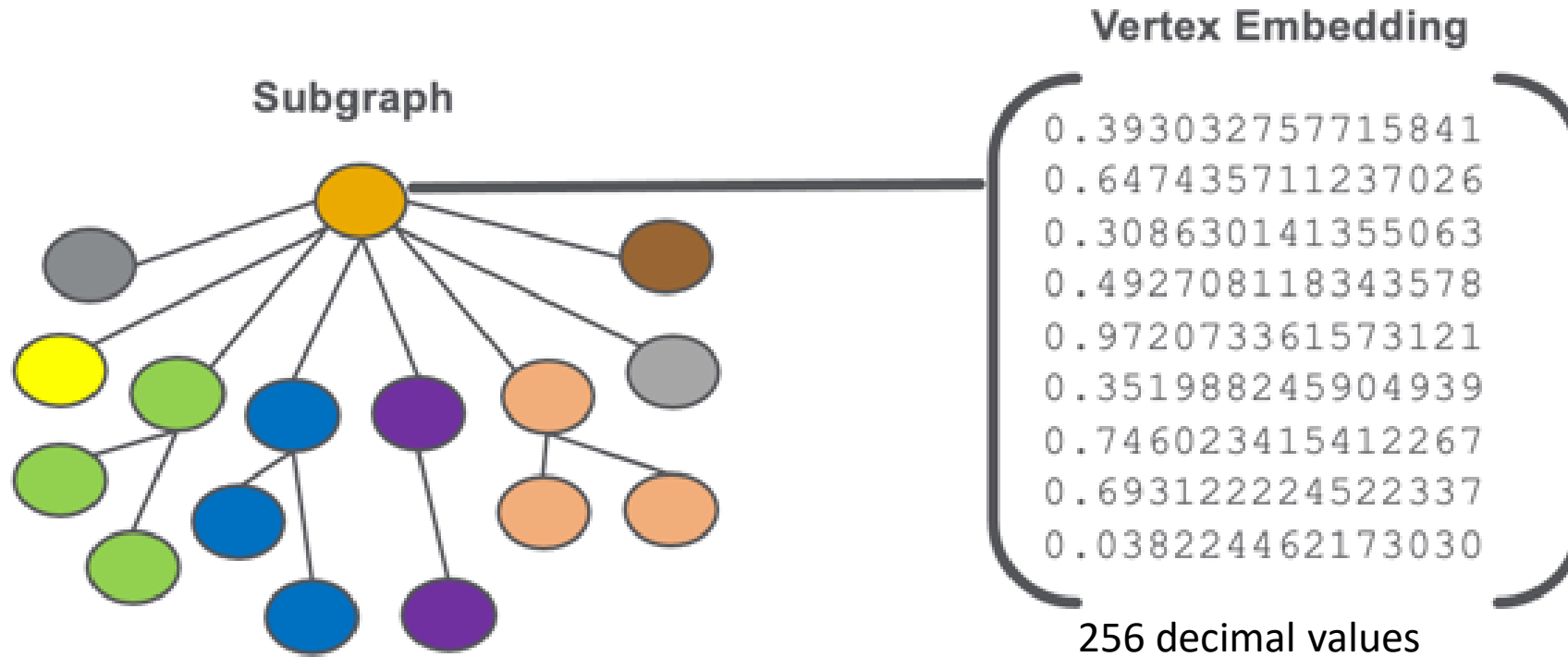
How many family practice physicians are there in Minneapolis that are taking new Medicare patients?

Text Embedding

0.393032757715841  
0.647435711237026  
0.308630141355063  
0.492708118343578  
0.972073361573121  
0.351988245904939  
0.746023415412267  
0.693122224522337  
0.038224462173030

128 decimal values

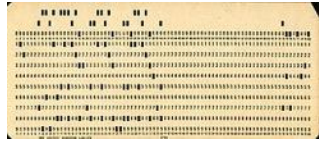
# Graph Embedding



A Representation of a Physician, Member, Customer etc.

# GenAI is about **Comparison** (not counting)

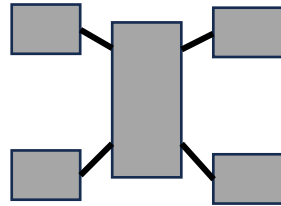
## Old Way: Counts and Amounts



COBOL Flat Files



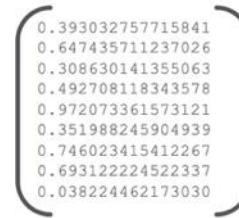
Spreadsheets  
RDBMS Tables



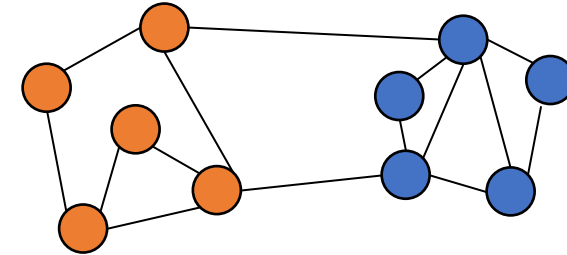
OLAP Cube

- How do we store knowledge for fast counting?
- Tables and OLAP Cubes

## New Way: Comparison



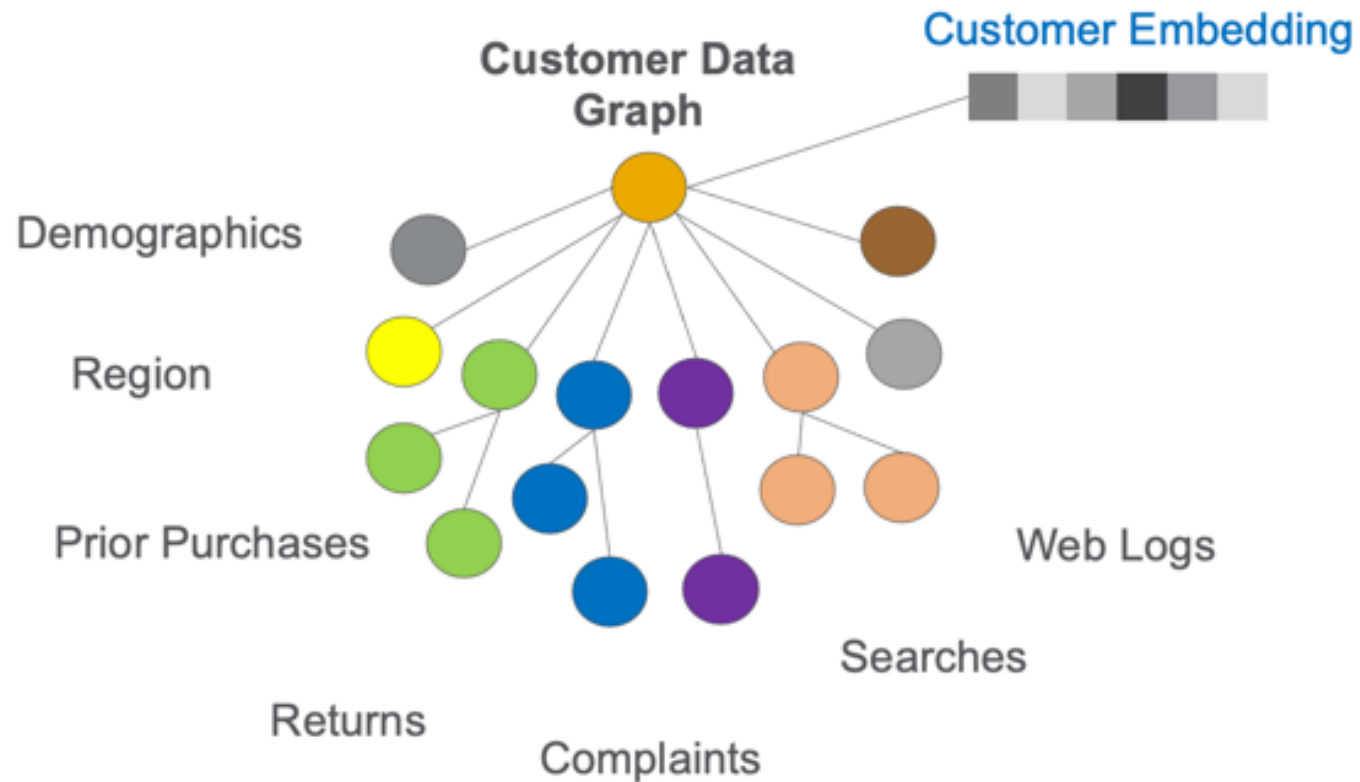
Embedding



Similarity

- How do we store knowledge for comparison?
- Graphs and Embeddings

# Customer Embeddings

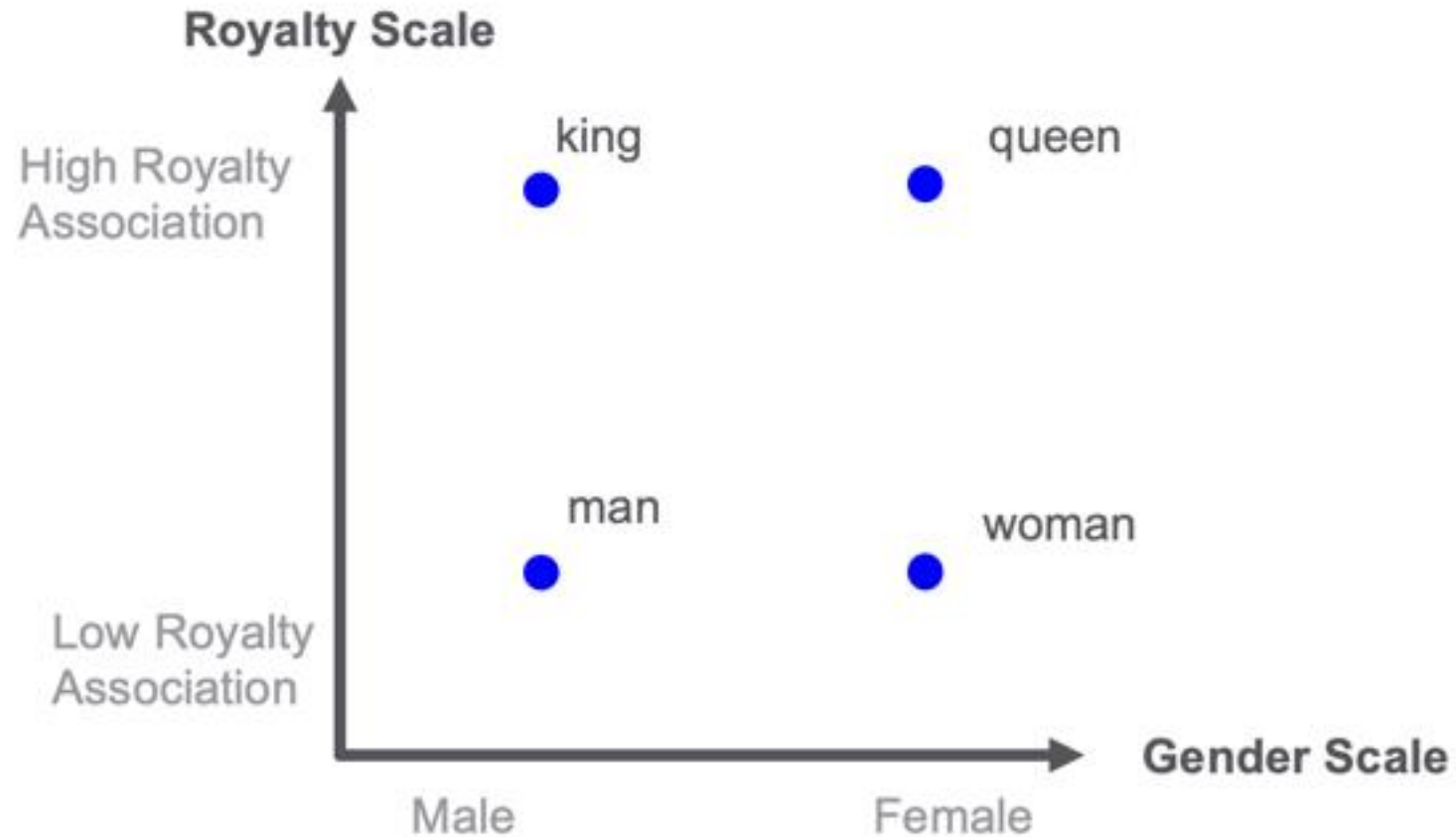


Comparing customers depends on storing their properties and linking customer touchpoints

Building embeddings is the domain of graph machine learning

# Distance Between Words

---



Manually rating 40K common English words is too expensive!

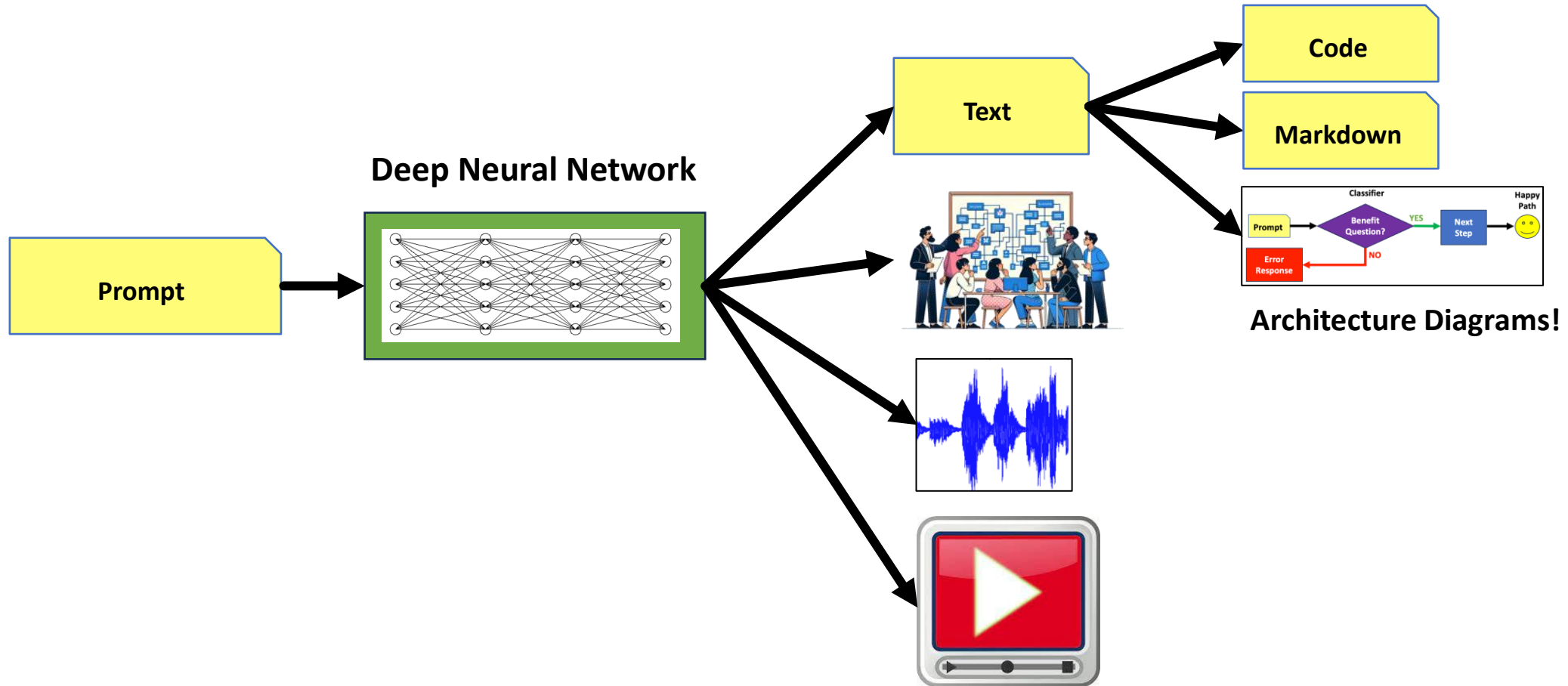
# Key Question: What Do You **Never** Want to Compare?

---

- Customers
- Products
- Competitors
- Competitors products
- Product comments
- Words
- Sentences
- Paragraphs
- Documents
- Images
- Workflows
- Code
- Queries
- APIs
- Action items assigned during a meeting
- Frequently asked questions
- Applications on the desktop
- Documents in a knowledge base
- Spreadsheets
- Reports
- Graphics
- PowerPoint slides
- Virtual machine configurations
- Errors in a log file

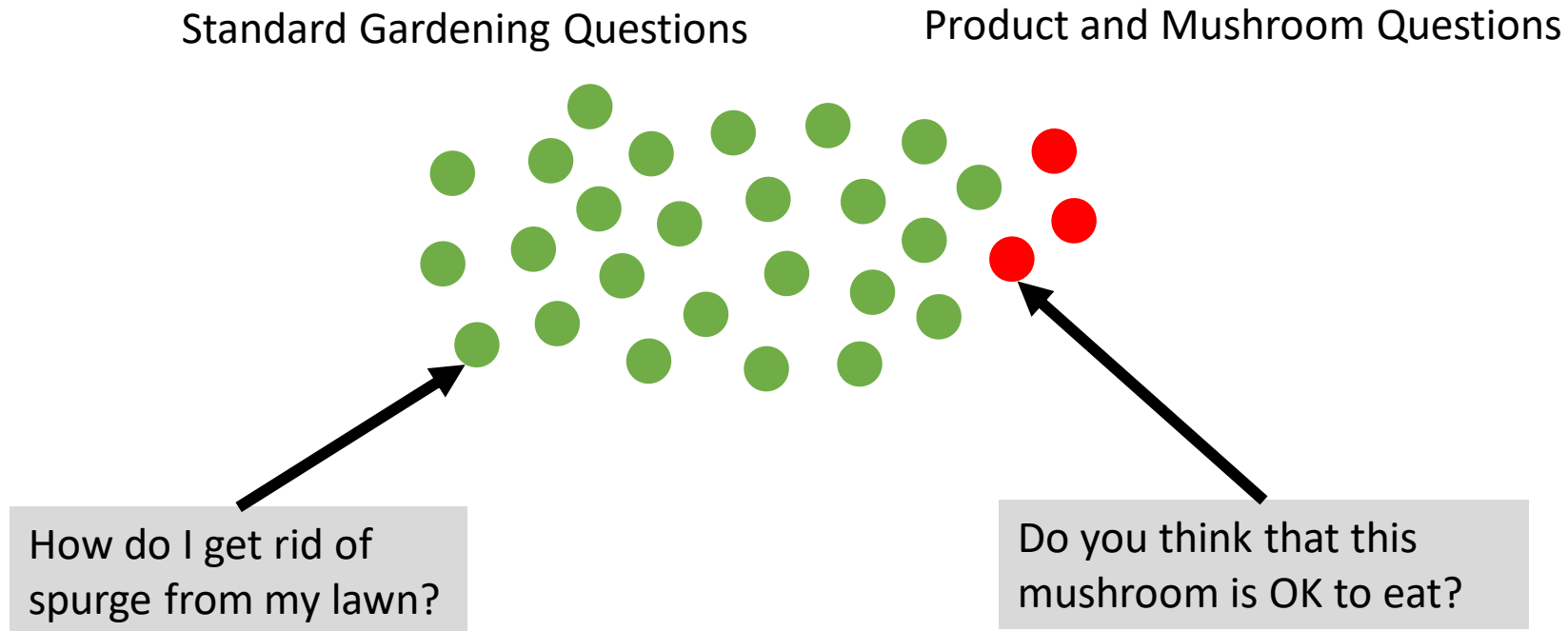
Answer: Our answer: none of the above!

# Multi-Modal Generative AI



Generative AI includes the creation of many content types including images, sound, music, videos, code, markdown, and even architecture diagrams

# Example: Gardening Question Chatbot



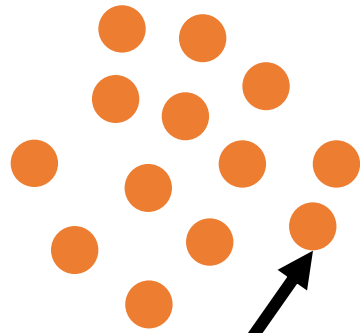
- You **can** answer standard questions about gardening, trees, shrubs, and lawns, etc.
- You **cannot** answer questions about commercial products or mushrooms



# Example: Benefits Chatbot

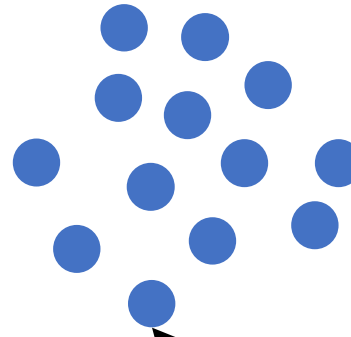
---

Benefit Questions



Is acupuncture covered by my plan?

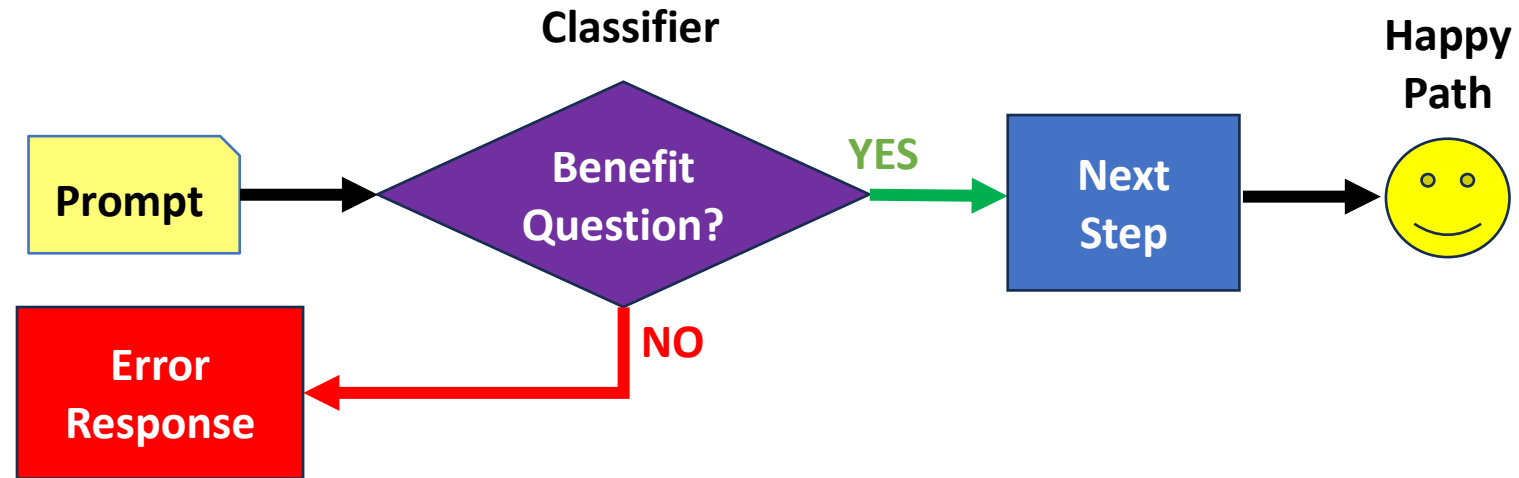
Medical Advice Questions



Do you think that acupuncture is effective?

- You can only answer questions about what is **covered** in your health plan
- You must **refuse** to answer questions about medical advice

# Sample Workflow

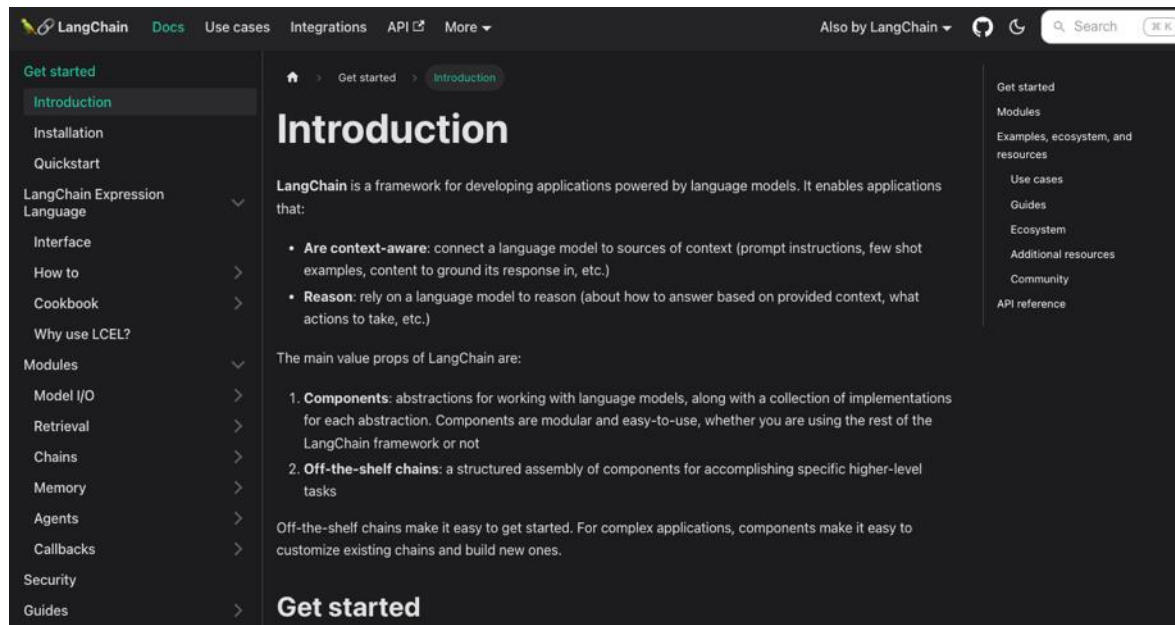


Architecture is about what boxes we have in our toolchest and how they are connected

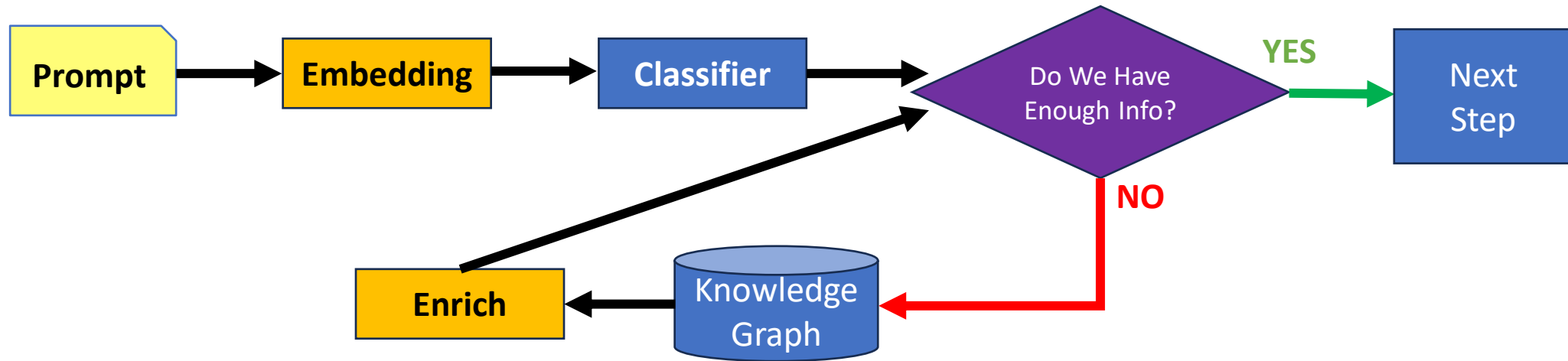
# LangChain



- Chains of Language Models
- Allows developers to quickly configure workflows that integrate LLMs



# Workflows, LangChain and Prompt Enrichment



- LangChain is a software system for building workflows where the components are often large-language models

# ChatGPT Advanced Data Analytics

Generate and Execute Python Code

Generate and Execute Python Code

life-expectancy.csv  
Spreadsheet

Create a data dictionary for this file.

Alright, let's start by examining the contents of the file. I'll load the first few rows of the dataset to understand its structure.

Finished working Show work

It appears there was an issue reading the file. The error suggests a discrepancy in the number of fields in certain lines, which may indicate irregular formatting or a mix of multiple data structures in the file.

To better understand the issue, I'll inspect the beginning of the file. This way, I can determine if there's a specific pattern or any irregularities present.

Finished working Show work

The file appears to start with some metadata, including the data source and the last updated date. The actual data begins on the fifth line, with columns indicating the country name, country code, indicator name, indicator code, and life expectancy values from 1960 to 2022.

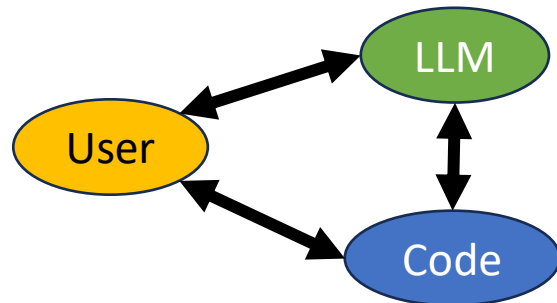
Loops of:

1. Generate code
2. See results of the data profiling Python code
3. Pick the next most similar workflow (that may also have Python code)

# Hundreds of Cooperating Agents

## Today

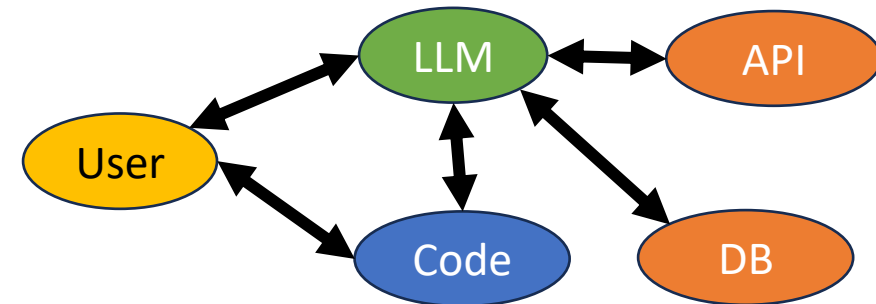
- Single prompt
- Single Large-language model (LLM)
- LLMs generate code
- Executed code generates prompts
- Example:  
ChatGPT Advanced Data Analytics



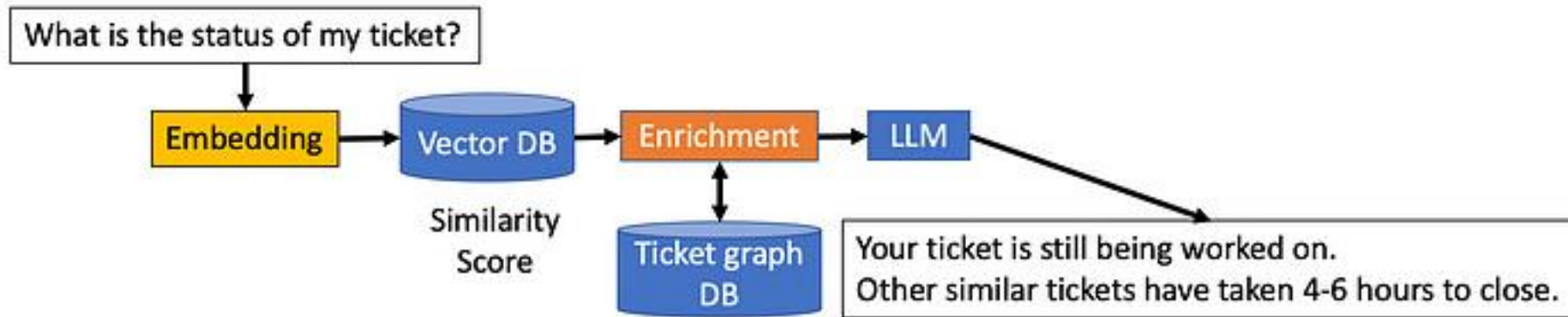
3-Agents  
System

## Next Year

- Hundreds of agents interacting inside your organization
- Agents calling services and databases
- LLMs are only limited by your company's ability to describe the APIs

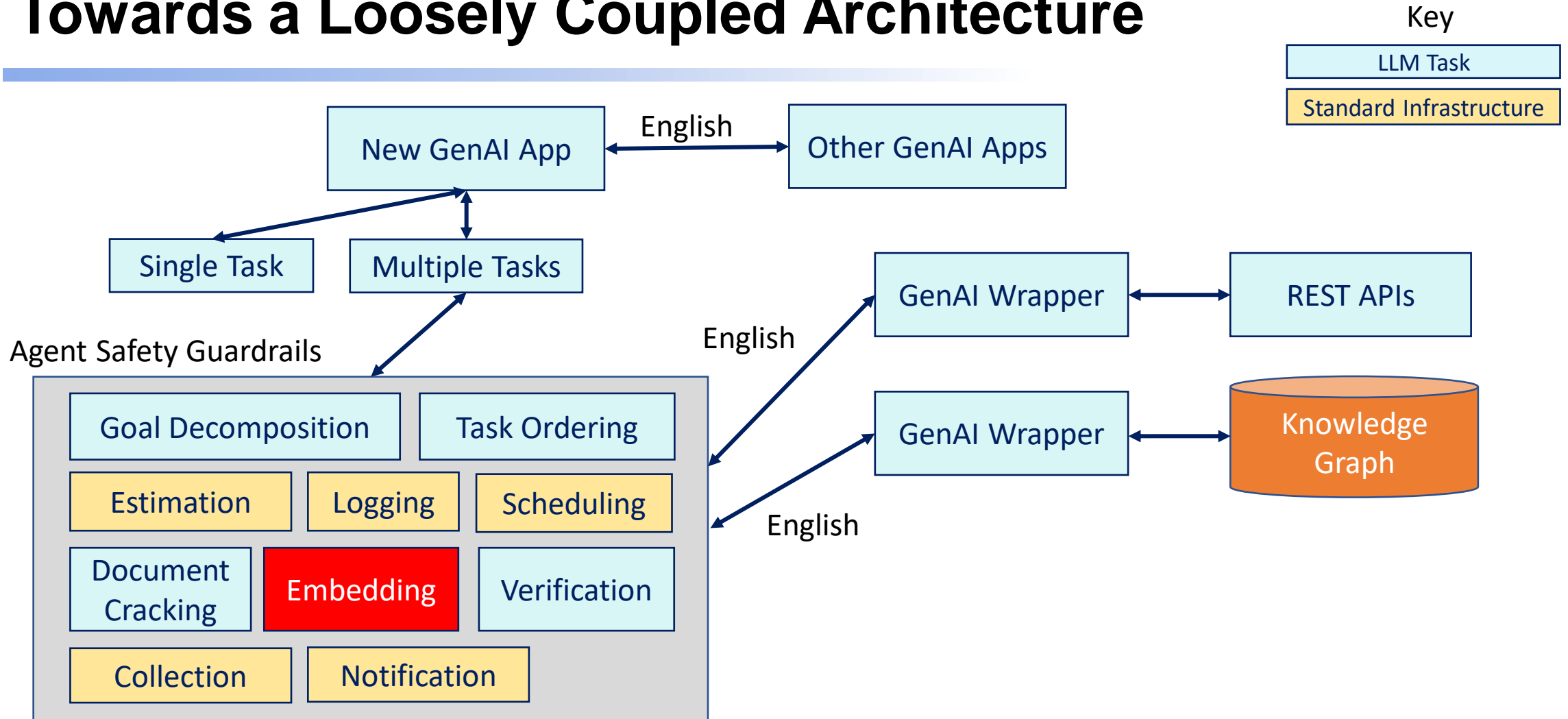


# Helpdesk Chatbot



- Similarity can tell you if other questions needed a query executed to get the right results
- The system must understand context (who you are and your context)
- Example:
  - Docker problems on a PC – average of over 2 hours to fix
  - Docker problems on the Mac – average less than 20 minutes to fix

# Towards a Loosely Coupled Architecture



Goal: define loosely coupled resilient architecture where clearly defined interfaces allow components to be swapped in and out as technology changes and new contracts are negotiated.

- Architecture



# Finding The Simplest Architecture

---

Complexity has a cost

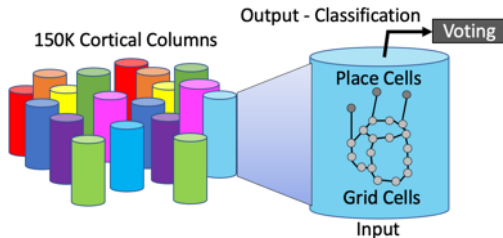
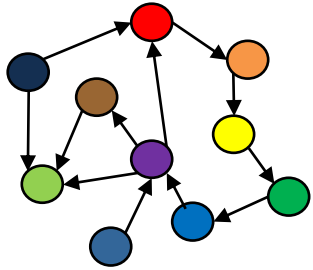
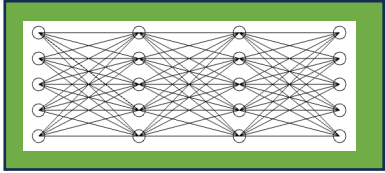
Can we get by with a simpler architecture?

The **cost** of  
building a model



The of **benefits** of  
using the model

# Comparing Knowledge Representations



1. Large-Language Model (Deep Neural Network)  
Works well with current CPU and GPU hardware but not a model of the world around us
2. Knowledge Graph (Nodes and Edges with Properties)  
Our best attempt at creating detailed models of the world around us with efficient traversal without JOINS
3. Reference Frames (used by the human brain)  
The way that human intelligence is **really** represented in the neocortex

# GenAI Strategy References

---

- The Generative Turn:

<https://dmccreary.medium.com/the-generative-turn-for-tech-strategy-502390aafb76>

- The Jellyfish and the Flatworm

<https://iianalytics.com/community/blog/the-jellyfish-and-the-flatworm-a-story-about-ai-strategy>

# Questions

---

# Thank You!

Site for this presentation: <https://dmccreary.github.io/genai-arch-patterns>

Dan's LinkedIn: <https://www.linkedin.com/in/danmccreary>

Steve's LinkedIn: <https://www.linkedin.com/in/steven-peterson-7928995/>

Dan's Blog: <http://medium.com/@dmccreary>