

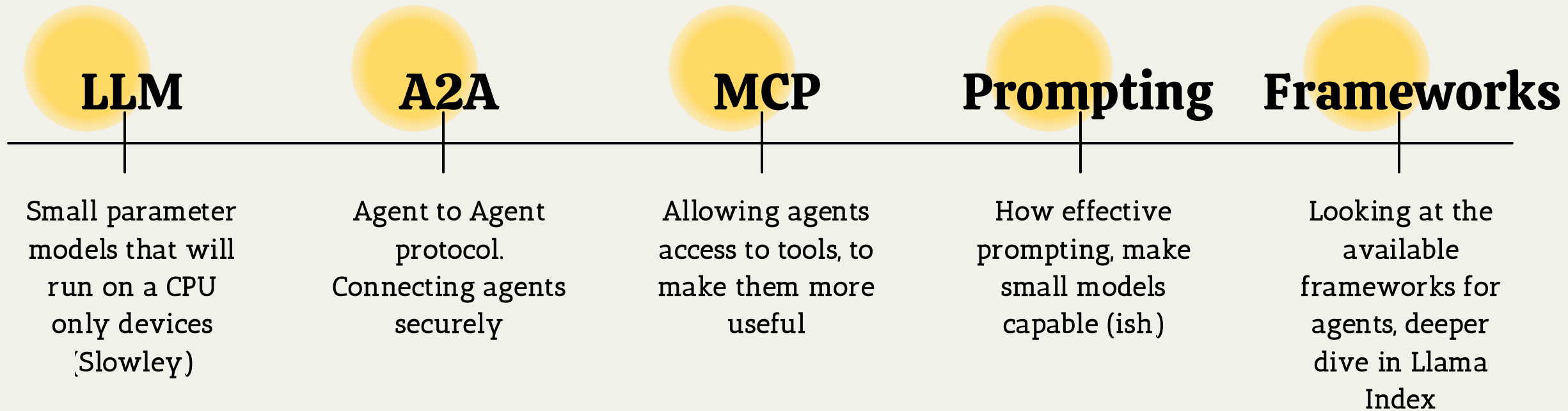
The little server that could: AI Agents on a Raspberry Pi



Brad Webb – Junior Vibe Coder

The little server that could: AI Agents

What we will talk about



LLM's

When people think LLM, what do we think of? OpenAI or Anthropic?

What about Ollama?

Ollama is an open source project, that hosts open source models, that you can run locally on your own hardware

Have a spare gaming PC at home? Can run models locally, when not playing Dota of course.



Discord



LLM's

Although the models hosted on the large labs are extremely powerful and have all the features (eg tools).

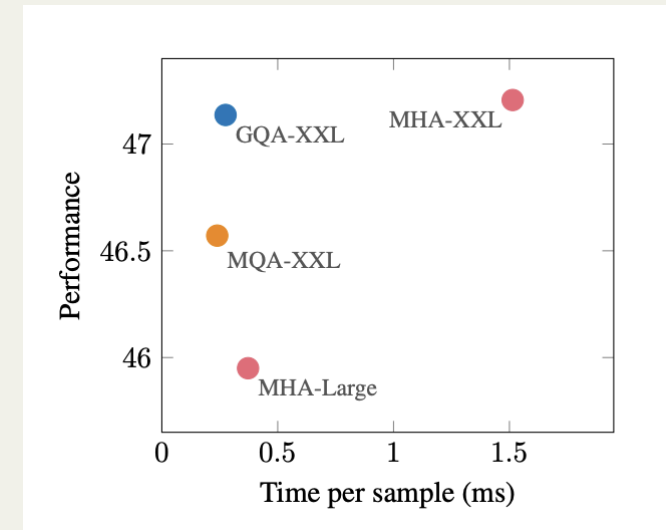
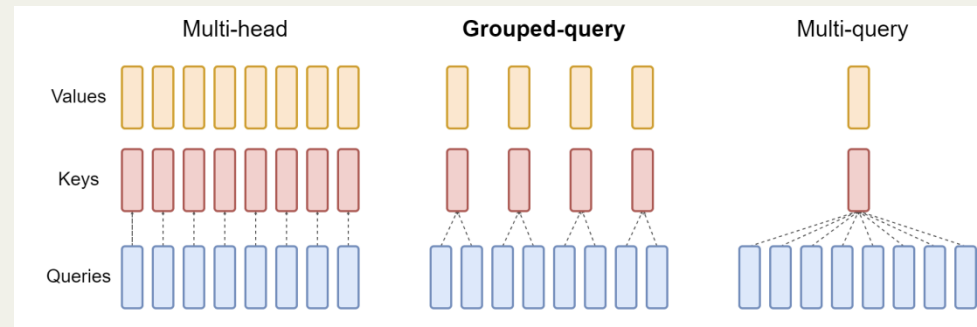
Local models are informative; lets you explore locally and offline

What model did I use? TinyLlama for inference, Nomic embedding

Why? Small, modern architecture, runs on raspberry pi CPU, good enough

<https://github.com/jzhang38/TinyLlama>

TinyLlama specs



1.1 billion parameter, trained on 16 A100-40G GPUs (3 till tokens)

Uses Llama 2 architecture (grouped query attention)

Fast and accurate training mechanism. Good small model performance, lower cost

Downside is it may limit expressiveness capture of inputs (Syntax, semantics, positional information)

Ref: GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints

[Joshua Ainslie](#), [James Lee-Thorp](#), [Michiel de Jong](#), [Yury Zemlyanskiy](#), [Federico Lebrón](#), [Sumit Sanghai](#)

Ref:
<https://x.com/pommedeterre33/status/1671263789914677248>

TinyLlama vs GPT-4

Can run on ARC infrastructure

Extremely fast (Compared to other models, can run on cpu)

Comparatively small, 1.1 billion params

Little models go Brrrrr

Model	Params	Tokens Trained	GPUs Used	Duration	Est. Cost
TinyLlama	1.1B	3 trillion	16× A100-40GB	90 days	\$25K–\$100K
ChatGPT-4	~1T – 1.7T est	10–20 trillion	Tens of thousands	Months	\$50M–\$100M+



Qdrant database



Lightweight vector DB, think sqlite but for vectors (super simple)

Good for POC's and development work, better databases for production

Has a simple python client, surprised at ease of use

Open source

```
[  
    array([-0.1115,  0.0097,  0.0052,  0.0195, ...], dtype=float32),  
    array([-0.1019,  0.0635, -0.0332,  0.0522, ...], dtype=float32)  
]
```

Nomic embed specs

High performance embedder, ideal for small devices

736 vectors (Less math to find similarity)

Used to create embeddings, database handles similarity

Hindsight: Qdrant encourages fastembed as part of their libraries,
should of used that (seems simpler)



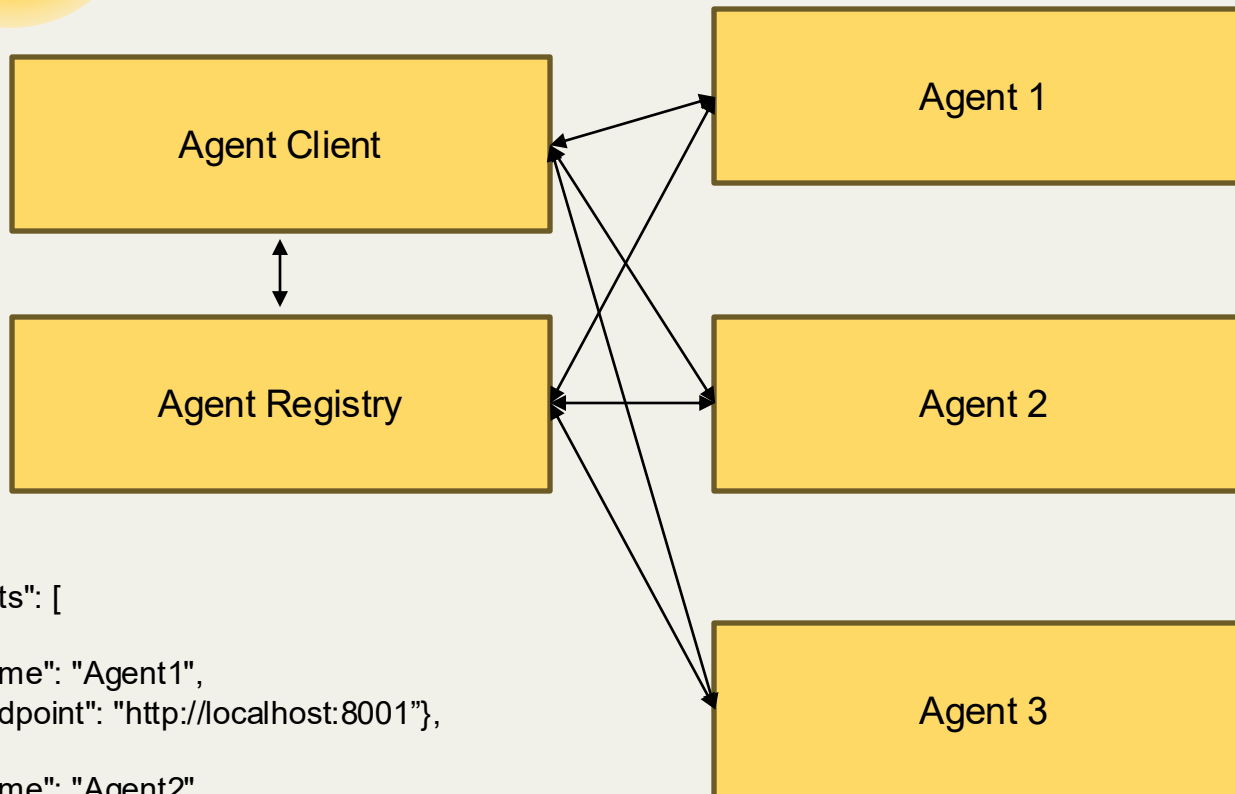
A2A

I fell for the marketing hype, A2A is a cool concept. Not production ready.

Allows for agent discovers with Agent Cards, so LLM's can autonomously explore agents to interact with

Designed for Autonomous agents, rather than explicit design flows (Deterministic design)

A2A Design: Agent card



EG:

```
{
  "agents": [
    {
      "name": "Agent1",
      "endpoint": "http://localhost:8001"},
    {
      "name": "Agent2",
      "endpoint": "http://localhost:8002"},
    ...
  ]
}
```

EG:

```
{
  "name": "Agent1",
  "description": "Credential Issuer Agent for issuing credentials.",
  "url": "http://localhost:8001",
  "version": "1.0.0",
  "capabilities": {
    "streaming": true,
    "stateTransitionHistory": true
  },
  "authentication": {
    "schemes": ["Bearer"],
    "credentials": "*****"
  },
  "defaultInputModes": ["application/json"],
  "defaultOutputModes": ["application/json"],
  "skills": [
    {
      "id": "issue-cred",
      "name": "Issue Credential",
      "description": "Issues credentials to connected agents.",
      "tags": ["credential", "issuer", "identity"],
      "examples": ["Issue a digital student ID"]
    }
  ]
}
```



A2A : The good

Agent Cards: Lets LLM's explore what agents are available, their 'skills', 'url' and 'capabilities'

Authentication: Allows for OpenAPI auth structure, essential for enterprise agents. (0 trust architecture)

Tags: Useful in a world where we have 100's or 1000's of agents.
Easier to group (We are not there yet)

Its more a flashy narrative, than
a product... xD
(Andrea Rettaroli 2025)

A2A : The bad

It's as undercooked as raw chicken

It's just a flask app, not serverless, requires HTTP connections, adds a lot of complexity to agent design

Centralised discovery server. Server breaks, all agents break

Maybe useful when autonomous agentic flows are used, currently a solution looking for a problem

All online docs/blog posts are written like a marketing press release.
Limited developer adoption (I suspect, Google is creating hype)





MCP

MCP (With HTTP) seems that it is the technology, that will make agents smart (Adding tools)

Standard protocol, allowing agents/llms to connect to a binary world (Functions, api's ect)

Many 'Official' servers available (HTTP adoption limited) allowing for connection to services (Check out AWS MCP!!!!)

Could be used as a wrapper for current API's, agents -> api's



MCP: The Good

Good standard protocol for connecting tools. OpenAI + AWS have their methods; this is open standard for the industry.

Allows a binary world, to be connected to LLM's

Has MCP inspector, a playground for testing connections and functionality



MCP: The Bad

Initially built around installing servers on your own machine (stdio), rather than hosted applications/cloud, http

Running local MCP servers proposes a MASSIVE security

HTTP is becoming the new standard, very few useful servers from providers available for this (Think Github, Microsoft). *Atlassian

How cool are we that LLM's could interact with systems (eg SQL), do we need to explore observability and a backtrack ability?



Prompting

Agents are a constraint on LLM, with clear instructions, generally tools available, autonomous design (micro ecosystem for llm)

You need to give each agent a clear prompt as to their action, what they can do, things not to do

Constraining their actions, will improve accuracy

With clear prompts even small models can perform well, one of the benefits of a larger model is you can be lazier with prompts



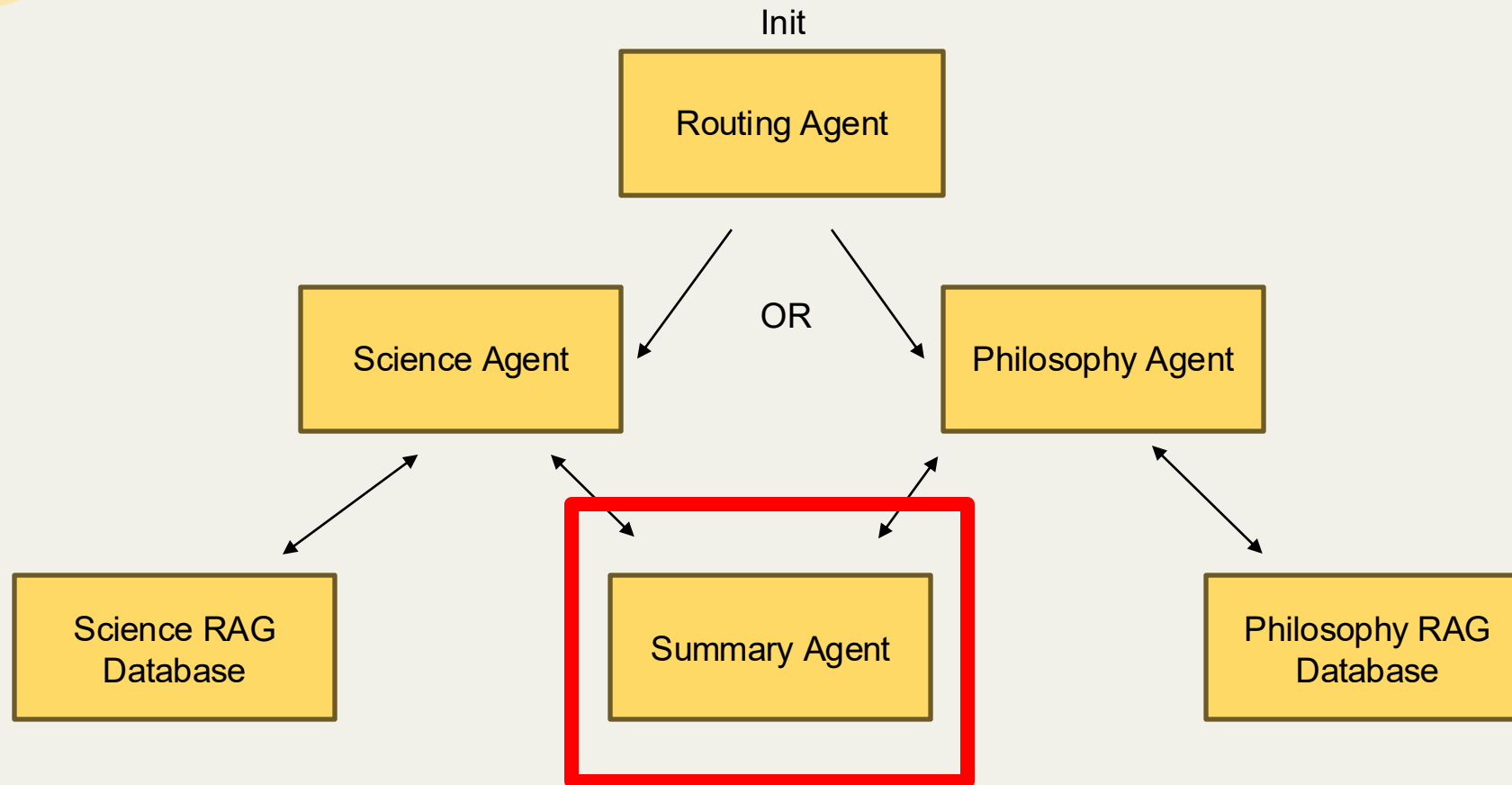
The hard learnings of software

Small models are like a sugared up, tired 3yr old's

My battle with the summary agent.....

Init -> Science Agent -> Science Rag database -> Science Agent -> Summary Agent ->
Philosophy Agent -> Philosophy Rag Database -> Philosophy Agent ->

How the debating agents interact



```

class SummaryAgent():
    def __init__(self):
        self.name = 'SummaryAgent'
        self.system_prompt = 'You are a helpful summary agent, you return summaries of the conversation no longer than 5 words'
        self.agent = FunctionAgent(
            name=self.name,
            description='You are a summary agent, you send back short and consise summaries',
            llm=Ollama(model="tinylama"),
            system_prompt=self.system_prompt
        )

    def run(self, user_query, context={}):
        try:
            print(self.agent.system_prompt)
            # Combine user query, database context, and additional context
            response = self.agent.llm.complete(f"{user_query}. ")
            print('Summary')
            print('='*20)
            print(response)
            return response
        except Exception as e:
            print(f"[ERROR] SummaryAgent encountered an error: {e}")
            return "Error occurred while processing the query."

```

5 Word Summary

Based on the given text, the following is a brief summary of the debate over free will and consciousness in the context of philosophy:

Free Will: Philosophers debate whether individuals possess autonomous control over their actions or are subject to external forces, such as genetic and environmental conditions. The debate intersects with questions about self-identity, personal identity, and consciousness. Some philosophers argue that free will is not entirely distinct from our brain and physical structures, while others suggest that the self is an embodied "conscious mind" that exists within these structures. Free Will: Philosophers also explore the relationship between free will and consciousness as the subjective experience of determining one's own actions. The debate intersects with neuroscience, which has enabled research into the nature of consciousness and the brain-body connection.

```

class SummaryAgent():
    def __init__(self):
        self.name = 'SummaryAgent'
        self.system_prompt = 'You are a 5 word summary agent, summarize to 5 words only'
        self.agent = FunctionAgent(
            name=self.name,
            description='You are a summary agent, you send back short and consise summaries',
            llm=Ollama(model="tinyllama"),
            system_prompt=self.system_prompt
        )

    def run(self, user_query, context={}):
        try:
            print(self.agent.system_prompt)
            # Combine user query, database context, and additional context
            response = self.agent.llm.complete(f"{user_query}. ")
            print('Summary')
            print('='*20)
            print(response)
            return response
        except Exception as e:
            print(f"[ERROR] SummaryAgent encountered an error: {e}")
            return "Error occurred while processing the query."

```

5 Word Summary

In summary, the philosophical debate on free will and consciousness has far-reaching implications for understanding moral responsibility, personal autonomy, and the nature of human existence. Compatibilism and libertarianism are two major stances held by philosophers in this debate, with each arguing that free will refers to freedom to choose between alternative courses of action based on initial conditions, while Libertarianism argues that choices are always freely made due to deterministic conditions. These debates have far-reaching implications for our understanding of moral responsibility and personal autonomy, emphasizing the importance of conscious choice in shaping behavior, environmental conditioning and biologic factors influencing decision-making. The evolutionary theory and behaviorism of the 19th century challenged traditional notions of human autonomy, while recent advances in neuroscience and gene-tics have further complicated this picture, revealing a complex interplay of biologic, psychological, and environmental factors that influence decision-making.

```

from llama_index.core.agent.workflow import FunctionAgent
from llama_index.llms.ollama import Ollama

class SummaryAgent():
    def __init__(self):
        self.name = 'SummaryAgent'
        self.system_prompt = 'You are a summary agent, you summarise to a very short answer'
        self.agent = FunctionAgent(
            name=self.name,
            description='You are a summary agent, you send back short and consise summaries',
            llm=Ollama(model="tinyllama"),
            system_prompt=self.system_prompt
        )

    def run(self, user_query: str):
        try:
            response = self.agent.llm.complete(f"return a ***5 WORD SUMMARY*** of this query: {user_query}. ")
            print('5 Word Summary')
            print('='*20)
            print(response)
            return response
        except Exception as e:
            print(f"[ERROR] SummaryAgent encountered an error: {e}")
            return "Error occurred while processing the query."

```

```
=====
5 Word Summary
=====
```

Based on this query, below is a 5-word summary of it:

In essence, the concept of free will refers to an individual's ability to make choices and choose oneself despite external influence and circumstances beyond their control. Philosophers debate the meaning and existence of free will for understanding personhood and consciousness, with different perspectives on its existence in other beings beyond humanity. The issue of intentionality, causality, and objectivity helps explain how we perceive and justify our choices, while compatibilism suggests that free will exists even when prior causes determine motivations. In contrast, other philosophers maintain that true free will requires the ability to choose otherwise in a given situation, which is difficult to reconcile with modern science.

```

from llama_index.core.agent.workflow import FunctionAgent
from llama_index.llms.ollama import Ollama

class SummaryAgent():
    def __init__(self):
        self.name = 'SummaryAgent'
        self.system_prompt = 'You are a summary agent, you summarise to a very short answer'
        self.agent = FunctionAgent(
            name=self.name,
            description='You are a summary agent, you send back short and consise summaries',
            llm=Ollama(model="tinyllama"),
            system_prompt=self.system_prompt
        )

    def run(self, user_query: str):
        try:
            response = self.agent.llm.complete(f"return a ***5 WORD SUMMARY*** of this query: <query>{user_query}</query>. ")
            print('5 Word Summary')
            print('='*20)
            print(response)
            return response
        except Exception as e:
            print(f"[ERROR] SummaryAgent encountered an error: {e}")
            return "Error occurred while processing the query."

```

PhilosophyAgent has a response

5 Word Summary

This query is comprised of five words that summarize the main topic of the given passage: free will, autonomous individuality, consciousness, self-construction, and morality, personal agency, and human existence. These themes have been at the forefront of philosophical debate for centuries and continue to provoke discussion among philosophers and thinkers alike. The idea of whether humans have genuine autonomy in making choices is a complex one that raises questions about morality, personal agency, and the nature of consciousness itself. Ultimately, this debate continues to fascinate and intrigue philosophers and thinkers alike, as it highlights the richness and complexity of the human experience.



Frameworks

AWS CDK – AWS native agents on Bedrock (Typescript)

LlamaIndex – Python library

Langchain – Python library

CrewAi – Honorable mention



AWS CDK Agent

```
// Create the agent
this.agent = new bedrock.CfnAgent(this, 'Agent', {
  agentName: props.agentName,
  foundationModel: props.foundationModel,
  instruction: Instruction,
  agentResourceRoleArn: this.agentExecutionRole.roleArn,
  actionGroups: actionGroups.length > 0 ? actionGroups : undefined,
  autoPrepare: true,
});
```




Llama Index Agent

```
self.supervisor_agent = FunctionAgent(  
    name="SupervisorAgent",  
    llm=Ollama(model="tinyllama"),  
    description="Moderates debates and summarizes outcomes.",  
    system_prompt="You are a neutral debate moderator.",  
    can_handoff_to=["PhilosophyAgent", "ScienceAgent"]  
)
```



Langchain Agent

```
prompt = ChatPromptTemplate.from_messages(  
    [  
        (  
            "system",  
            "You are a helpful assistant. Make sure to use the tavily_search_results_json tool for inf  
        ),  
        ("placeholder", "{chat_history}"),  
        ("human", "{input}"),  
        ("placeholder", "{agent_scratchpad}"),  
    ]  
)  
  
# Construct the Tools agent  
agent = create_tool_calling_agent(llm, tools, prompt)
```



Why LlamaIndex

Simpler design, beginner friendly, simpler syntax

More focused on RAG and class-based design, works out of the box

Better for simpler/standard implementations. Less flexible than Langchain

Langchain has Langsmith (monitoring/evaluation), Langserve (API)

Llama Index fits better inside my definition of Software development
(I understand its implementation better)

```

import os
from llama_index import VectorStoreIndex, SimpleDirectoryReader
from llama_index.llms import OpenAI

class RAGApplication:
    def __init__(self, data_dir: str, model_name: str = "gpt-3.5-turbo"):
        self.data_dir = data_dir
        self.model_name = model_name
        self.llm = OpenAI(model=model_name)
        self.index = None
        self.query_engine = None

    def load_documents(self):
        """Load documents from a directory."""
        print(f"Loading documents from {self.data_dir}...")
        reader = SimpleDirectoryReader(self.data_dir)
        documents = reader.load_data()
        return documents

    def build_index(self):
        """Build the vector index from the loaded documents."""
        print("Building vector index...")
        documents = self.load_documents()
        self.index = VectorStoreIndex.from_documents(documents, llm=self.llm)
        self.query_engine = self.index.as_query_engine()

    def ask(self, question: str) -> str:
        """Query the index with a question."""
        if not self.query_engine:
            raise ValueError("Query engine is not initialized. Call build_index() first.")
        print(f"Asking: {question}")
        response = self.query_engine.query(question)
        return response.response

```

```

def interactive_loop(self):
    """Run an interactive QA loop."""
    print("RAG app is ready. Type 'exit' to quit.")
    while True:
        user_input = input("Ask a question: ")
        if user_input.lower() in {"exit", "quit"}:
            print("Goodbye!")
            break
        try:
            answer = self.ask(user_input)
            print("Answer:", answer, "\n")
        except Exception as e:
            print("Error:", e)

```

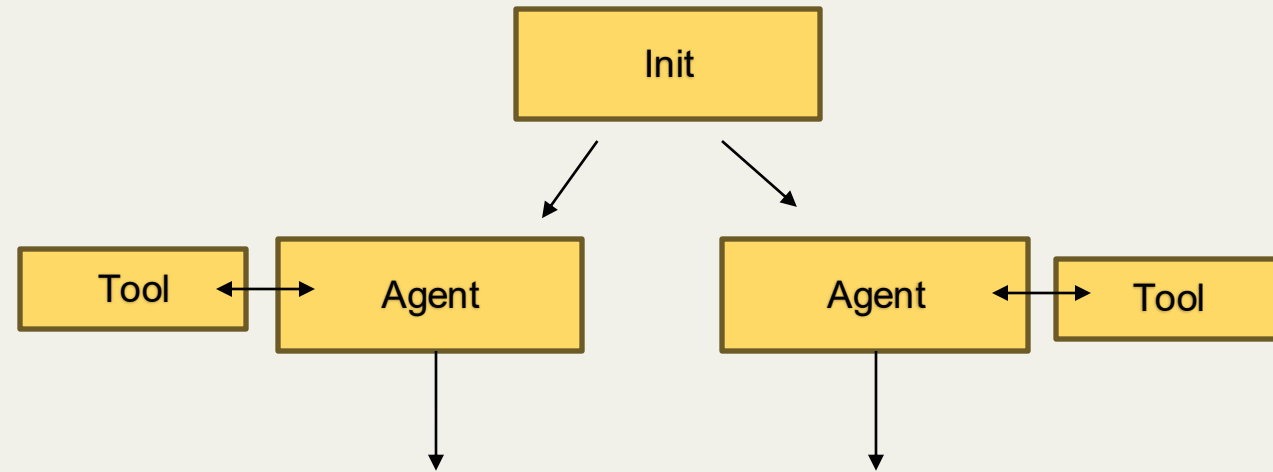
```

if __name__ == "__main__":
    # Optional: set your API key if not already set
    os.environ["OPENAI_API_KEY"] = "your-openai-key"

    app = RAGApplication(data_dir="data")
    app.build_index()
    app.interactive_loop()

```

Function Agents



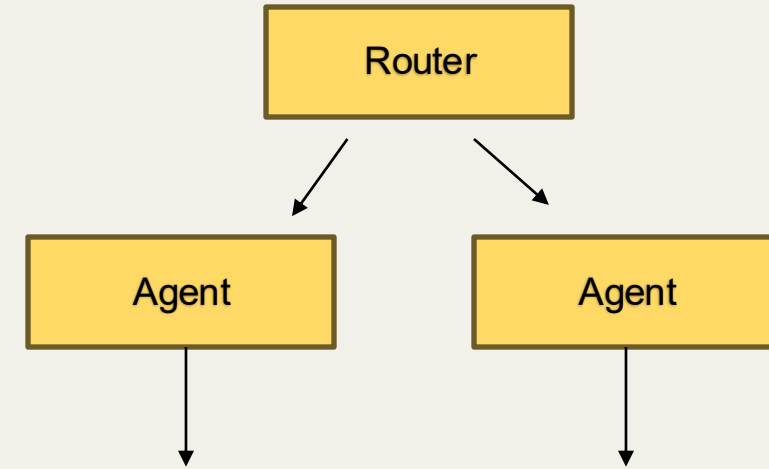
Function agents are simple agents that can call tools

Input task, give tools (mcp, tool, api call) allows the task to be executed (w/ additional context if needed)

Deterministic, acts in a much more constrained way

****Think 'Do an action'****

Common patterns: Router



Acts as the traffic controller, delegating tasks to subsequent Agents

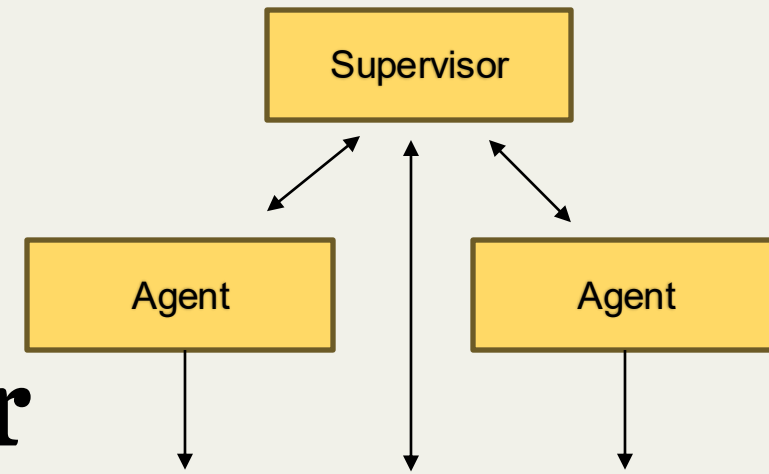
Uses rules/NLP to route to different agents

Minimal state kept, designed to be simple and resilient

Generally: Takes a text input, parses/interprets it, passes request off to another agent to perform the task

More a one-way street (Similar to a DAG aka Data Pipeline)

Common patterns: Supervisor



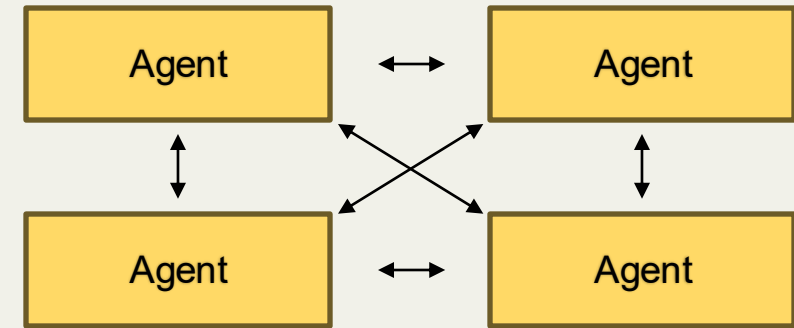
Oversees and monitors complex agents/ambiguous agent structures, the brain of the agents

Can make decisions, delegate sub tasks to other agents, resolve conflict and manage an overall goal

Holds state, can refer to previous agent interactions and override actions

More a two way street, Supervisor is the manager

reAct Agents



reAct agents are reasoning and action agents

Has autonomy over decisions, can skip steps, open ended design, can loop over ideas within the agent (Multiple LLM calls)

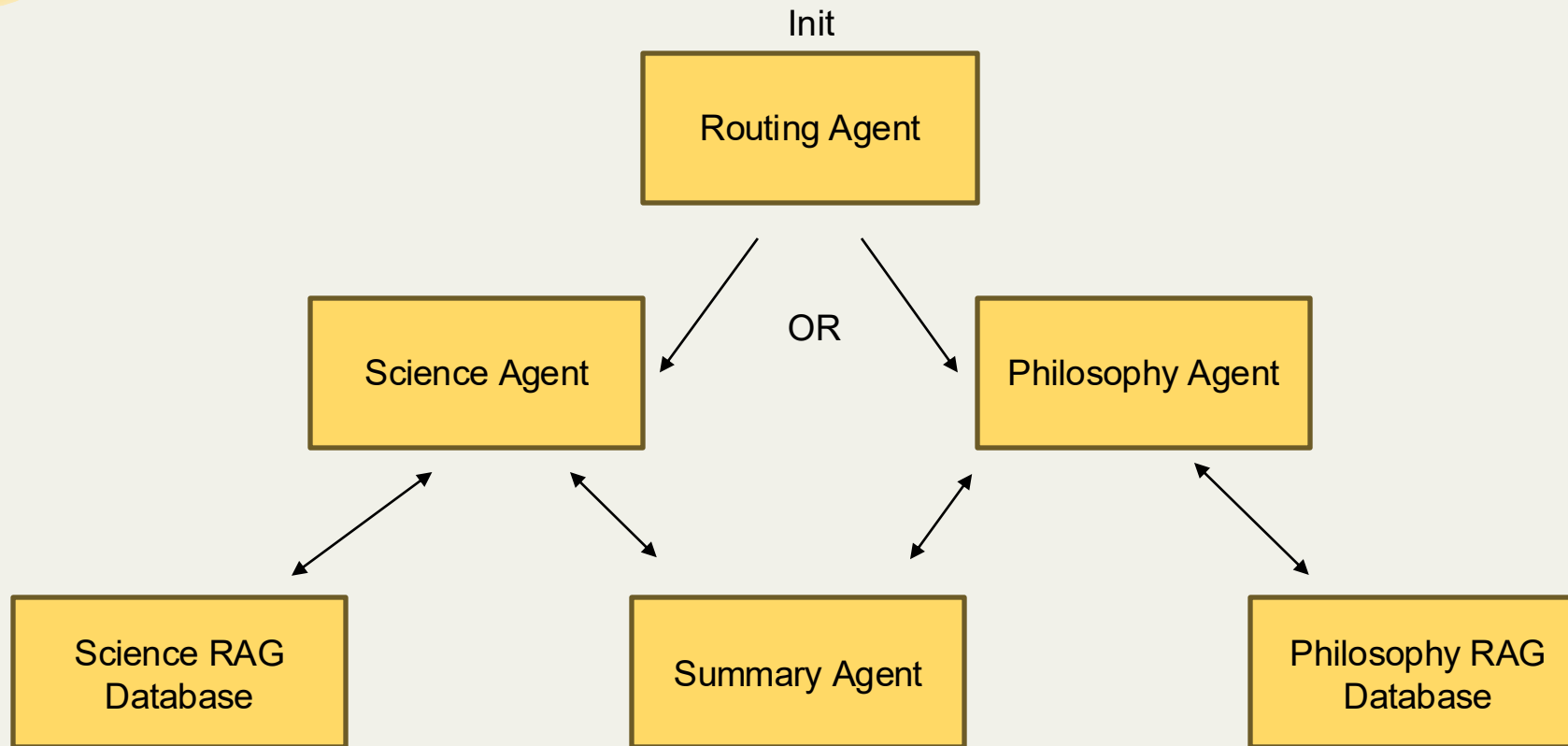
Good for exploratory or loosely defined processes


A2A might be useful here, allowing a central registry, for reAct agents to explore/understand the ecosystem

****Think explore an idea, solve the problem****

Init -> Science Agent -> Science Rag database -> Science Agent -> Summary Agent ->
Philosophy Agent -> Philosophy Rag Database -> Philosophy Agent ->

How the debating agents interact





Live demo – Agent debate

Wish my luck – Live examples are risky



Live demo – MCP SQL server (Bonus)

Wish my luck – Live examples are risky

Need to connect to the internet on this one, so not 100% offline



What I learnt

Cannot vibe code any of these libraries or ideas, they are so new the models hallucinate. I had to read source code ☹

The top end Lab models (Claude and ChatGPT) are soo good. They are making us lazy, but with some prompts and data they are powerful.

Agents will start appearing more soon. The tech is almost there, and a lot of things could implemented. Even if not full autonomous

These models smash compute, chased Timeouts across the project